UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Construção de um Ambiente de Desenvolvimento de Software baseado em um Sistema de Gerência de Workflow e outros Produtos Comerciais

por

Carlos Michel Betemps

Dissertação submetida à avaliação,
como requisito parcial, para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Roberto Tom Price
Orientador

Porto Alegre, março de 2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Betemps, Carlos Michel

Construção de um Ambiente de Desenvolvimento de Software baseado em um Sistema de Gerência de Workflow e outros Produtos Comerciais / por Carlos Michel Betemps - Porto Alegre: PPGC da UFRGS, 2003.

162 f.:il

Dissertação de Mestrado – Universidade Federal de Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2003. Orientador: Price, Roberto Tom.

1. Ambientes de Desenvolvimento de Software 2. Sistemas de Gerência de *Workflow* 3. Produtos de Prateleira 4. COTS 5. Processo de Software. 6. Processo Unificado Rational. 7. *Exchange 2000 Server*. 8. *Internet*. 9. *Active Server Pages*. I. Price, Roberto Tom. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitora: Profª . Wrana Panizzi

Pró-reitor de Ensino: Prof. José Carlos Ferraz Hennemann

Pró-Reitora Adjunta de Pós-Graduação: Profª . Jocélia Grazia

Diretor do Instituto de Informática: Prof. Philippe Oliver Alexandre Navaux

Coordenador do PPGC: Prof. Carlos Alberto Heuser

Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

Dedico este trabalho ao amor de minha vida, minha esposa Carla.

“…
A spirit breaking free
One little victory
The greatest act can be
One little victory
...”

Neil Peart - **Rush**

Agradecimentos

Em primeiro lugar, gostaria de agradecer a Deus pela minha vida e por me manter na direção certa nos difíceis, mas compensadores, passos desta longa caminhada chamada ***Vida***.

Gostaria de agradecer aos meus pais, Carlos e Irene, pelo incentivo nas horas difíceis, pela minha criação e por minha vida. Aos meus irmãos, Daniel e Maurício, pelas horas indispensáveis de descontração. Aos meus avós Domingos e Tereza, valeu a força padrinhos. A minha vó Nair, desculpe a ausência e obrigado por me dar um exemplo de fé. Aos meus sogros, Carlos e Tereza, obrigado pela ajuda nas horas que a gente precisa. A minha tia emprestada, Maria, valeu pelos almoços gostosos.

Gostaria de agradecer a Carla, minha esposa. Desculpe os momentos em que estive ausente e obrigado por me ajudar com minhas dificuldades, me dar incentivo e por me aturar quando o stress era alto. “Te amo Amor”.

Não poderia deixar de agradecer a todo pessoal da UFRGS, que se esforçam para manter o nosso ambiente de trabalho um local agradável de se estar. Agradeço a instituição UFRGS, por ser este verdadeiro exemplo de organização e respeito.

Agradeço ao professor Roberto Tom Price pela sua orientação. Sua experiência e conhecimentos foram essenciais para a realização deste trabalho. Valeu pelas “broncas”.

Agradeço ao professor Marcelo Soares Pimenta pela ajuda e pela oportunidade de trabalharmos juntos. Se este trabalho tem um Co-orientador, “o cargo é teu”.

Agradeço a CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo apoio financeiro, sem o qual, este trabalho jamais teria sido realizado.

Agradeço ao pessoal da UFPel, professores e alunos, pela ajuda e amizade.

Agradeço ao pessoal do futebol da UFRGS. Os nossos jogos, as piadas, a conversa jogada fora, as “brigas”, a “tiração de sarro”, os almoços no “Le´RU” e a nossa lista de e-mail vão ficar na memória o resto da minha vida.

Finalizando, foi extremamente gratificante para o autor deste trabalho ter tido a oportunidade de cursar o Programa de Pós Graduação em Computação da Universidade Federal do Rio Grande do Sul, que, durante o tempo de andamento deste trabalho, comprovou a sua fama de um dos melhores cursos de pós-graduação do Brasil.

Sumário

Lista de Abreviaturas e Siglas **8**

Lista de Figuras **9**

Lista de Tabelas **11**

Resumo **12**

Abstract **13**

1 Introdução **14**

1.1 Objetivos e Resultados Esperados **16**
1.2 Estrutura do Trabalho **18**

2 Ambientes de Desenvolvimento de Software **19**

2.1 Repositórios de ADS **21**
2.2 Funcionalidades de Hipertexto e Hiperdocumentos nos ADS **22**
2.3 Modelagem e Execução de Processos em ADS-CP **24**
2.4 Requisitos necessários para um ADS **25**
2.5 Resumo do Capítulo **28**

3 Sistemas de Gerência de *Workflow* **30**

3.1 Padrões da WFMC para Sistemas de Gerência de *Workflow* **31**
3.2 Groupware **34**
3.3 Tipos de *Workflow* **36**
3.4 Características Típicas dos Sistemas de Gerência de *Workflow* **37**
3.5 Limitações dos Sistemas de Gerência de *Workflow* **39**
3.6 Alguns Sistemas de Gerência de *Workflow* **40**
3.6.1 Ultimus Workflow 40
3.6.2 Oracle *Workflow* Cartridge 41
3.6.3 MQSeries *Workflow* da IBM 43
3.6.4 TIBCO - InConcert *Workflow* 44
3.6.5 BizFlow da HandySoft 45
3.6.6 Microsoft Exchange 2000 Server 47
3.7 Comparação entre os SGW Apresentados **54**
3.8 Resumo do Capítulo **56**

4 Processo de Software **57**

4.1 Processos de Software **58**
4.1.1 Processo ***OPEN*** 58
4.1.2 Processo ***OOSP*** 59

4.1.3 Processo ***RUP*** 60
4.2 Processo Unificado Rational - RUP 62
4.2.1 Alguns Conceitos Associados ao RUP 63
4.2.2 Workflows e Artefatos Associados 65
4.3 Requisitos para um Processo de Software 72
4.4 Limitações do RUP 74
4.5 Aplicação de SGW na Modelagem e Execução de Processos de Software 74
4.6 Resumo do Capítulo 76

5 Arquitetura Baseada em SGW e Ferramentas Comerciais para Construção de ADS 78

5.1 Uso de Produtos Comerciais de Prateleira (COTS) como Componentes de Construção 78
5.2 Construção de ADS utilizando SGW e Outras Ferramentas Comerciais 80
5.3 Alguns Trabalhos Relacionados 81
5.4 Arquitetura Proposta 84
5.5 Vantagens e Desvantagens de Arquiteturas Baseadas na Aplicação de Ferramentas Comerciais como Unidades de Construção 90
5.6 Comparação com os Trabalhos Relacionados 93
5.7 Adequação da Arquitetura Proposta aos Requisitos definidos na Seção 2.495
5.8 Modelagem e Execução de Processos de Software 97
5.9 Resumo do Capítulo 98

6 Protótipo Implementado 99

6.1 Configurações do *Microsoft Exchange 2000 Server* 99
6.1.1 Organização das Pastas Públicas Utilizadas pelo WOSDIE 99
6.2 Itens do Exchange 2000 e suas Respectivas Propriedades 103
6.2.1 Item Atividades 103
6.2.2 Item Equipe 105
6.2.3 Item Projeto 106
6.2.4 Item Equipe de Revisão 108
6.2.5 Item Solicitação de Alteração 108
6.2.6 Item de Configuração de Atividade 108
6.2.7 Item de Configuração de Ferramenta 109
6.2.8 Item de Configuração de Papéis 109
6.2.9 Relacionamento entre os Itens do Exchange utilizados no WOSDIE 110
6.3 Processo de Software do Ambiente 110
6.3.1 Modelando Processos de Software no *Microsoft Workflow Designer for Exchange 2000 Server* 113
6.4 Ativação de Ferramentas a Partir do WOSDIE 117
6.5 Artefatos Gerados 119
6.6 Arquitetura do Protótipo 120
6.7 Interfaces de Acesso do WOSDIE 122
6.8 Casos de Uso do WOSDIE 132
6.9 Avaliação do WOSDIE 133
6.9.1 Tecnologia PML 135
6.9.2 Arquitetura do ADS-CP 140
6.9.3 Experiências 142
6.10 Possíveis Extensões para o WOSDIE 143

6.11 Resumo do Capítulo 145

7 Conclusões 147

7.1 Trabalhos Futuros 151
7.2 Considerações Finais 153

Referências 154

Lista de Abreviaturas e Siglas

AD	Active Directory – Diretório Ativo
ADO	ActiveX Data Objects – Objetos de Dados ActiveX
ADS	Ambientes de Desenvolvimento de Software
ADS-CP	Ambientes de Desenvolvimento de Software Centrados em Processo
API	Application Programming Interface – Interface de Programação de Aplicação
ASP	Active Server Pages - Páginas de Servidor Ativas
BD	Banco de Dados
CDO	Collaboration Data Objects – Objetos de Dados de Colaboração
CMM	Capability Maturity Model
COTS	Commercial off-the-shelf – produtos de prateleira.
CSCW	Computer Supported Cooperative Work – Trabalho Cooperativo Auxiliado por Computador
DTD	Document Type Definition – Definição de Tipo de Documento
E2K	Exchange 2000 Server
HAD	Heterogêneos, Autônomos e Distribuídos
IT	Information Technology – Tecnologia da Informação
MAPI	Messaging Application Programming Interface – Interface de Programação de Aplicação de Mensagens
OLE	Object Linking and Embedding – Vinculação e Incorporação de Objetos
PML	Process Modelling Languages - Linguagens de Modelagem de Processos
PSEE	Process-centred Software Engineering Environments
SEE	Software Engineering Environments
SGW	Sistema de Gerência de Workflow
UML	Unified Modeling Language – Linguagem de Modelagem Unificada
URL	Uniform Resource Locator – Localizador de Recurso Uniforme
WFMC	Workflow Management Coalition – Coalisão de Gerenciamento de Workflow
WfMS	Workflow Management System
WPDL	Workflow Process Definition Language – Linguagem de Definição de Processo de Workflow
WSS	Web Storage System – Sistema de Armazenamento Web
XML	eXtensible Markup Language – Linguagem de Marcação Extensível

Lista de Figuras

FIGURA 3.1 - Modelo de Referência de SGW [HOL 95] 32
FIGURA 3.2 - Estrutura Genérica de Produtos de *Workflow* [HOL 95] 33
FIGURA 3.3 - Oracle *Workflow* Builder [ORA 98] 42
FIGURA 3.4 - Buildtime do MQSeries IBM [MQS 2000] 44
FIGURA 3.5 - Interface com o usuário do InConcert [TIB 2000] 45
FIGURA 3.6 - Process Designer do BizFlow [BIZ 2000] 46
FIGURA 3.7 - Interface do Microsoft *Workflow* Designer for Exchange 2000 Server 50
FIGURA 3.8 - Função VBScript para Criação de Mensagens de E-mail 53
FIGURA 3.9 - Exemplo de Consulta SQL sobre os Itens de "Atividades" 53
FIGURA 4.1 - Ciclo de Vida dirigido por contrato do processo OPEN [AMB 2001] 59
FIGURA 4.2 - Ciclo de Vida do Processo *OOSP* [AMB 2001] 60
FIGURA 4.3 - RUP - Estrutura do Processo [RAT 98] 61
FIGURA 4.4 - Menu Principal do RUP 63
FIGURA 4.5 - Relacionamentos entre Atividade, Artefato, Papel e Trabalhador 64
FIGURA 4.6 - Workflow de Modelagem de Negócios [RAT 2001] 65
FIGURA 4.7 - Workflow de Requisitos [RAT 2001] 66
FIGURA 4.8 - Workflow de Análise e Projeto [RAT 2001] 67
FIGURA 4.9 - Workflow de Implementação [RAT 2001] 67
FIGURA 4.10 - Workflow de Teste [RAT 2001] 68
FIGURA 4.11 - *Workflows* e Artefatos Associados [ERI 2000] 69
FIGURA 4.12 - Workflow de Implantação [RAT 2001] 69
FIGURA 4.13 - Workflow de Gerenciamento de Configuração e Alteração [RAT 2001] 70
FIGURA 4.14 - Workflow de Gerenciamento de Projeto [RAT 2001] 71
FIGURA 4.15 - Workflow de Ambiente [RAT 2001] 71
FIGURA 5.1 - Arquitetura Típica de um ADS-CP [FUG 96] 84
FIGURA 5.2 - Arquitetura Proposta para ADS 86
FIGURA 5.3 - Modelo de Componentes para ADS 88
FIGURA 5.4 – Diagrama de Seqüência mostrando alguns relacionamentos dinâmicos dos Componentes da Arquitetura 89
FIGURA 6.1 - Hierarquia de Pastas do WOSDIE 100
FIGURA 6.2 - Exemplo de Código para a Criação de uma Pasta Pública 104
FIGURA 6.3 - Modelo de Classes Relacionando os Itens Utilizados no WOSDIE 110
FIGURA 6.4 - Processo de Software do Ambiente. 112
FIGURA 6.5 - Modelagem no Workflow Designer de parte do Modelo de Processo 115
FIGURA 6.6 – Campos para Modelagem no *Workflow Designer* 116
FIGURA 6.7 - Função JavaScript para Ativação do Rational Rose 117
FIGURA 6.8 - Funções *JavaScript* utilizando o Controle *ActiveX* LaunchinIE [WHI 2002a] 118
FIGURA 6.9 - Registrando URLs seguras para o Controle *ActiveX* LaunchinIE [WHI 2002a] 119
FIGURA 6.10 - Modelo de Componentes do WOSDIE 121
FIGURA 6.11 - Interface de *Logon* no Ambiente 122
FIGURA 6.12 - Interface Principal do Ambiente 123
FIGURA 6.13 - Interface de Projetos Cadastrados 123
FIGURA 6.14 - Edição dos Planos de Projeto 124
FIGURA 6.15 - Interface de Acompanhamento de Projetos e Integração de Artefatos 124
FIGURA 6.16 - Interface de Acompanhamento de um Projeto Individual 125
FIGURA 6.17 - Interface de Acesso aos Artefatos Concluídos de um Projeto 125
FIGURA 6.18 - Interface das Listas de Atividades 126
FIGURA 6.19 - Edição de Atividades e *Upload* de Artefatos 127
FIGURA 6.20 - Interface de Cadastro de Solicitações de Alteração 127
FIGURA 6.21 - Interface de Edição de Cadastro de Solicitações de Alteração 128

FIGURA 6.22 - Equipe Cadastrada no Ambiente....128
FIGURA 6.23 - Edição das Informações de um Usuário....129
FIGURA 6.24 - Interface de Configurações do WOSDIE....129
FIGURA 6.25 - Interface de Edição de Papéis (Funções)....130
FIGURA 6.26 - Interface das Atividades do Ambiente....130
FIGURA 6.27 - Interface de Edição das Atividades do WOSDIE....131
FIGURA 6.28 - Ferramentas Cadastradas no Ambiente....131
FIGURA 6.29 - Cadastro de Ferrametas do WOSDIE....132
FIGURA 6.30 - Casos de Uso do WOSDIE....133
FIGURA 6.31 - Documento DTD para um item de Atividade....144
FIGURA 6.32 - Exemplo de Documento XML referente a atividade "Elicitar Solicitação dos Interessados"....145
FIGURA 7.1 - Exemplo de *Script* para criação de documento XML....152

Lista de Tabelas

TABELA 3.1 - Comparação entre os SGW apresentados 55
TABELA 4.1 – Os Cinco Níveis de Maturidade do Modelo CMM [AMB 2001] 72
TABELA 4.2 – Comparativo dos Processos RUP, OPEN e OOSP em relação as Áreas-Chave de Processo do Modelo CMM [AMB 2001].....73
TABELA 5.1 - Comparativo entre a abordagem baseada na integração de ferramentas comerciais e construção por completo 91
TABELA 6.1 – *Explorer* de Registro de Formulários [MAR 2000] 102
TABELA 6.2 - Propriedades do Item de Atividades 105
TABELA 6.3 - Propriedades do Item de Equipe 105
TABELA 6.4 - Identificadores, Descrição e Papéis Associados às Atividades do Processo de Desenvolvimento.....106
TABELA 6.5 - Propriedades do Item de Projeto.....107
TABELA 6.6 - Propriedades do Item de Equipe de Revisão.....108
TABELA 6.7 - Propriedades do Item de Alteração.....108
TABELA 6.8 – Propriedades do Item de Configuração de Atividade.....109
TABELA 6.9 - Propriedades do Item de Configuração de Ferramentas.....109
TABELA 6.10 - Propriedades do Item de Configuração de Papéis.....109
TABELA 6.11 - Atividades e seus Gabaritos e Manuais de Orientação Associados [RAT 2001].....113
TABELA 7.1 – Quadro Comparativo entre as Abordagens de Construção utilizando Produtos COTS e Construção a partir do Zero.....150

Resumo

Este trabalho apresenta uma arquitetura para Ambientes de Desenvolvimento de Software (ADS). Esta arquitetura é baseada em produtos comerciais de prateleira (COTS), principalmente em um Sistema de Gerência de Workflow – SGW (*Microsoft Exchange 2000 Server* – E2K) - e tem como plataforma de funcionamento a *Internet*, integrando também algumas ferramentas que fazem parte do grande conjunto de aplicativos que é utilizado no processo de desenvolvimento de software.

O desenvolvimento de um protótipo (WOSDIE – *WOrkflow-based Software Development Integrated Environment*) baseado na arquitetura apresentada é descrito em detalhes, mostrando as etapas de construção, funções implementadas e dispositivos necessários para a integração de um SGW, ferramentas de desenvolvimento, banco de dados (WSS – *Web Storage System*) e outros, para a construção de um ADS.

O processo de software aplicado no WOSDIE foi extraído do RUP (*Rational Unified Process* – Processo Unificado Rational). Este processo foi modelado na ferramenta *Workflow Designer*, que permite a modelagem dos processos de *workflow* dentro do E2K. A ativação de ferramentas a partir de um navegador *Web* e o armazenamento dos artefatos produzidos em um projeto de software também são abordados.

O E2K faz o monitoramento dos eventos que ocorrem dentro do ambiente WOSDIE, definindo, a partir das condições modeladas no *Workflow Designer*, quais atividades devem ser iniciadas após o término de alguma atividade anterior e quem é o responsável pela execução destas novas atividades (assinalamento de atividades).

A arquitetura proposta e o protótipo WOSDIE são avaliados segundo alguns critérios retirados de vários trabalhos. Estas avaliações mostram em mais detalhes as características da arquitetura proposta e proporcionam uma descrição das vantagens e problemas associados ao WOSDIE.

Palavras-Chave: Ambientes de Desenvolvimento de Software, Sistemas de Gerência de *Workflow*, Processo de Software, Processo Unificado da Rational, *Exchange 2000 Server*, *Internet*, *Active Server Pages*.

TITLE: "BUILDING OF A SOFTWARE DEVELOPMENT ENVIRONMENT USING A WORKFLOW MANAGEMENT SYSTEM AND OTHERS COMMERCIAL PRODUCTS"

Abstract

This work presents a Software Development Environment (SDE) Architecture. This one is based on COTS products, mainly a Workflow Management System - WfMS (Microsoft Exchange 2000 Server – E2K), and run under the Internet platform, integrating a few development tools, used in the software development process.

The development of a prototype (WOSDIE – WOrkflow-based Software Development Integrated Environment) based on a presented architecture is described, showing the implementation steps, implemented functions and used devices to the integration of a WfMS, development tools, a database (WSS – Web Storage System), and others, in the building of a SDE.

The WOSDIE software process has been extracted from RUP (Rational Unified Process). This process was modeled in the Workflow Designer tool, which allow the software process modeling. The tools activation via browser web and the storage of software artifacts are too approached.

The E2K monitor the events in the WOSDIE environment. Based in the conditions modeled (and finished activities in the run time) in the Workflow Designer tool, activities are assigned (and initiated by) to the responsible workers.

The proposed architecture and the WOSDIE environment are assessed based in some features extracted of a few papers about software engineering environments. The architecture and WOSDIE assesses show in details the architecture features and describe the WOSDIE advantages and problems.

Keywords: Software Development Environments, Workflow Management Systems, Software Process, Rational Unified Process, Exchange 2000 Server, Internet, Active Server Pages.

1 Introdução

A produção de software de qualidade possui uma relação direta com a qualidade do processo de software utilizado para a produção destes produtos [FUG 2000]. A interação e troca de informações entre os membros dos grupos de desenvolvimento de software é um fator chave que determina o sucesso ou a falha de qualquer iniciativa de desenvolvimento [BAN 96]. A documentação sobre o processo de desenvolvimento, atividades, artefatos, papéis desempenhados, etc., fornece facilidade e informações para a realização das atividades de desenvolvimento de software [PER 92] [ORT 95]. O emprego de padrões de software (como padrões de projeto) facilita a atividade de desenvolvimento de software, por fornecer modelos de solução para os problemas mais comuns [GAM 2000]. O reuso de software é também uma maneira de aproveitar soluções já utilizadas em projetos anteriores para a resolução de problemas similares [GIM 99] [OIN 99].

Desenvolver software é um processo complexo. Pesquisadores e profissionais perceberam ao longo dos anos que na área de desenvolvimento de software a preocupação não é apenas criar linguagens de programação e ferramentas eficientes. Desenvolvimento de software é um esforço coletivo, complexo e criativo, e como tal, a qualidade de um produto de software depende muito das pessoas, da organização e dos procedimentos usados para criá-lo e liberá-lo para o uso [FUG 2000].

Vários Ambientes de Engenharia de Software (SEE - *Software Engineering Environments*), ou Ambientes de Desenvolvimento de Software (ADS), têm sido desenvolvidos para dar suporte e facilitar o desenvolvimento de software. Uma geração mais recente destes ambientes, chamados Ambientes de Engenharia de Software centrados em Processo (ADS-CP, em inglês: PSEE – *Process-centred Software Engineering Environments*), suportam a definição e execução de várias fases do processo de software; isto é obtido pela definição explícita dos procedimentos de cooperação e pelo suporte a sincronização e compartilhamento de dados entre os usuários dos ambientes [BAN 96].

ADSs estão se tornando uma realidade. Um número crescente de sistemas estão sendo apresentados e utilizados para processos de produção de software reais. Produtos e protótipos existentes são baseados em uma variedade de tecnologias e abordagens, tais como: linguagens e bancos de dados orientados a objetos, notações orientadas a estados, linguagens baseadas em regras e linguagens lógicas [FIN 94].

Ambientes como ADS-CPs e ADSs visam proporcionar aos seus usuários um conjunto de funcionalidades e facilidades que tornam a tarefa de desenvolvimento de software menos árdua e mais organizada, integrando técnicas, processos, métodos e ferramentas. Metodologias e processos de software podem ser aplicados e utilizados, mesmo de maneira personalizada, de uma forma automática através da adequação do ADS a um determinado processo de desenvolvimento de software e pela aplicação de metodologias de desenvolvimento específicas.

ADSs normalmente fornecem as ferramentas necessárias para o desenvolvimento de software (como compiladores, ferramentas de teste, editores de

diagramas, processadores de texto, prototipadores de interface, etc.), repositórios de dados (como banco de dados relacionais ou orientados a objetos, sistemas de arquivos, documentos XML, etc.), interfaces de acesso dos usuários aos dados do ambiente, dispositivos de ativação das ferramentas do ambiente, controle e gerência das atividades a serem desenvolvidas assim como a atribuição de atividades aos desenvolvedores, etc.

Outra área de intensa pesquisa nos últimos anos é a de Workflow e os Sistemas que suportam este conceito, os Sistemas de Gerência de Workflow (SGW). *Workflow* pode ser descrito como o movimento de documentos e tarefas através de um processo de negócio. Workflows podem ser seqüências de atividades de trabalho ou mesmo um conjunto complexo de processos, sendo que podem ocorrer processos executados concorrentemente e geralmente causando algum efeito na execução dos demais processos de acordo com os conjuntos de regras, rotas e papéis relacionados aos mesmos [DIC 97].

A Workflow Management Coalition (WFMC) define SGW como: "a automação de um processo de negócio, em todo ou parte, durante o qual documentos, informações ou tarefas são passadas de um participante para outro para realização de alguma ação, de acordo com um conjunto de regras procedimentais" [WMC 99].

Desenvolvimento de software pode ser considerado como um tipo especializado de processo de negócio, de maneira que documentos, informações e tarefas são passadas de um participante para outro de acordo com um processo de desenvolvimento [BAR 2000] e onde tarefas gerenciais - como alocação de atividades aos desenvolvedores, planejamento de prazos e contratação de pessoas - são realizadas pela equipe de gerência dos projetos [CHA 97] [BUS 98].

Se um processo de software é um tipo especializado de processo de negócio e Sistemas de Gerência de Workflow podem automatizar processos de negócio, então os Sistemas de Gerência de Workflow também podem automatizar processos de software.

Vários autores têm pesquisando a aplicação de SGW na construção de ADSs e na modelagem/execução de processos de software. Estes autores têm apresentado seus trabalhos e contribuições em muitas universidades, conferências, workshops, revistas e diversos eventos acadêmicos e industriais [ARA 99] [BAR 2000] [BAR 2000a] [BUS 98] [CHA 97] [CHA 97a] [CHA 99] [GRU 2000] [MAN 2001] [MAU 99] [OCA 98].

Processo de Software é um dos tópicos mais pesquisados dentro da Engenharia de Software. Um processo de software pode ser definido como o conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos que são necessários para a concepção, desenvolvimento, implantação e manutenção de um produto de software [FUG 2000] [GIM 94] [FIN 94].

Um processo de software descreve a interação entre pessoas e ferramentas na realização do trabalho envolvido no ciclo de vida do software. Um processo de software abrange o trabalho a ser realizado (as atividades), quem irá realizar (os desenvolvedores), o que será utilizado para realizar este trabalho (as ferramentas), o que será produzido (os artefatos) e quando e como este trabalho será realizado (o comportamento).

Muitos processos de software já foram propostos, como: OPEN [OPE 2002] [AMB 2001], OOSP [AMB 2001] [AMB 98] [PRO 2002] e RUP [RAT 2001] [RAT 98]. Dentre estes processos, o processo RUP vem chamando atenção pela suas características interessantes e sua aplicação em várias organizações, sua integração com

UML, sua motivação ao reuso e aplicação das melhores técnicas de desenvolvimento de software.

O processo RUP, segundo seus autores, integra as melhores práticas de desenvolvimento de software moderno em uma forma que é conveniente para uma grande faixa de projetos e organizações. As melhores práticas de desenvolvimento de software utilizadas pelo RUP são [RAT 98]:

(1) Desenvolver o Software Iterativamente;

(2) Gerenciar os Requisitos;

(3) Usar Arquiteturas baseadas em Componentes;

(4) Visualmente Modelar o Software;

(5) Verificar a Qualidade de Software;

(6) Controlar as Modificações do Software.

O processo RUP fornece todos os elementos necessários para o desenvolvimento de software, como informações sobre os seus *Workflows*, tipos de desenvolvedores (papéis), atividades, artefatos, manuais de orientação de atividades (*Guidelines*), ferramentas associadas, gabaritos de artefatos (*templates*), etc. [RAT 2001]. Estes elementos fornecem aos usuários meios de conduzir, gerir e avaliar o processo de desenvolvimento de aplicações.

Alguns trabalhos [ARA 99] [BUS 98] [GRU 2000] [LOO 99] [PAY 99] [YAK 99] [MEH 2000] [CAR 97] propuseram o uso, e estudaram os efeitos de utilização, de produtos comerciais de prateleira, ou produtos COTS (*Commercial off-the-Shelf*) (como Sistemas de Gerência de Workflow, Ferramentas de Desenvolvimento de Software, Navegadores *Web*, etc.) como componentes de construção de ADS e outros domínios. Araújo et. al. [ARA 99] apresentou um ambiente cuja a arquitetura, baseada na Web, integrava um servidor HTTP a um banco de dados orientado a objetos e utilizava um SGW (WebDeploy) para a modelagem, execução e monitoramento de processos de workflow (processos de software). Bussler [BUS 98] propôs a integração de SGW e Ferramentas de Gerência de Projetos para habilitar a execução controlada (planejada) de instâncias de workflow. Grundy et. al. [GRU 2000] descreve uma abordagem baseada em componentes (produtos comerciais como: editores, SGW, compiladores, etc.) para a construção de ambientes de engenharia de software.

1.1 Objetivos e Resultados Esperados

Neste trabalho é proposta a construção de Ambientes de Desenvolvimento de Software (ADS) a partir da utilização de Sistemas de Gerência de Workflow (SGW) como um dos componentes base. Também é proposta a integração de produtos comerciais de diferentes fabricantes (como ferramentas de produtividade do *Microsoft Office 2000* e outras, editores de modelagem como *Rational Rose 2000* [RAT 2001] e *TogetherSoft* [TOG 2001], ferramentas de programação como *Borland JBuilder* [BOR 2002], *Microsoft Visual Basic* e *Visual C++* [MIC 2001], ferramentas de teste como o *Rational TeamTest*, *TestManager* e *PureCoverage* [RAT 2001], e tantas outras

utilizadas no processo de construção de software), a utilização de páginas *Web* (programadas em ASP, *JavaScript* ou outro recurso necessário) como interface de acesso ao ambiente e aos documentos que constituirão o acervo de artefatos e informações gerados pelo ambiente, que por sua vez são armazenadas em um bancos de dados (que atua como um repositório dos dados gerados pelo ambiente). O ambiente é acessado pela *Internet* através de uma URL específica do servidor contendo o ADS.

O protótipo de experimentação deste trabalho foi desenvolvido sobre o *Exchange 2000 Server* (E2K) [EXC 2001]. Este é um servidor de colaboração que possui funcionalidades de gerência de *workflow* e que realiza o controle das atividades executadas no ambiente, faz a atribuição de atividades de acordo com o andamento dos projetos e possibilita a modelagem e execução dos processos de desenvolvimento de software como modelos de *Workflows*. O E2K é integrado ao sistema operacional *Windows 2000 Server* [MIC 2001]. Esta integração permite que o E2K utilize funcionalidades do *Windows 2000 Server* como o *Web Storage System* (WSS), que funciona como um banco de dados e que permite o armazenamento de informações relevantes do E2K, dos usuários do ambiente e dos *workflows* (processo de software) modelados e executados no E2K. O protótipo funciona através da plataforma *Internet*, permitindo que os usuários acessem o ambiente utilizando um navegador e um endereço IP. A interface de acesso às informações do ambiente é realizada através das páginas ASP e funções *JavaScript* que recuperam, atualizam e geram informações referentes aos projetos e ativam componentes e ferramentas externas. O ambiente realiza a ativação de ferramentas de desenvolvimento associadas às atividades através do servidor de automação do *Windows* e de controles *ActiveX* (componentes disponibilizadas gratuitamente na *Internet*). Também é realizada a geração dinâmica de hiperdocumentos contendo os artefatos produzidos nos projetos. Os artefatos gerados durante a realização das atividades podem ser carregados para o servidor através de operações de *upload*. Este tipo de operação é possível através da instalação no servidor *Web* - que atua como o *host* do protótipo - de um componente (*ActiveX*) que também é disponibilizado gratuitamente na *Internet*.

Os objetivos do trabalho podem ser resumidos nos seguintes tópicos:

- Utilização de produtos COTS como componentes de construção de ADS;
- Utilização de SGW como um dos componentes principais para a construção de ADS;
- Integração e aplicação de ferramentas de desenvolvimento de software na construção de ADS.
- Geração de hiperdocumento contendo os artefatos produzidos nos projetos;
- Disponibilização de *Links* dentro das páginas *Web* que servem como interface do ambiente, e que ativem ferramentas externas utilizadas para a realização das atividades;
- Proposta de uma arquitetura para construção de Ambientes de Desenvolvimento de Software (ADS);
- Definir os passos da modelagem de processos de software no SGW utilizado no protótipo do trabalho;
- Mostrar meios de ativação de ferramentas através de navegadores Web;

- Avaliar a utilização de produtos COTS (tal como SGW) na construção de ADS;
- Avaliar a aplicação de SGW na gerência de processos de software em um ADS;
- Avaliar a abordagem de construção baseada em produtos comerciais através da implementação de um protótipo.

1.2 Estrutura do Trabalho

Este trabalho está organizado da seguinte maneira:

- O capítulo 2 mostra os conceitos e características inerentes aos Ambientes de Desenvolvimento de Software. Também é definida uma lista de requisitos consideradas importantes para ambientes deste tipo.
- O capítulo 3 faz uma revisão sobre *Workflow* e Sistemas de Gerência de *Workflow*, mostrando alguns produtos existentes e caracterizando melhor o produto *Microsoft Exchange 2000 Server* e seus componentes associados.
- O capítulo 4 faz uma revisão simplificada sobre processo de software e descreve brevemente alguns processos propostos. O processo RUP é descrito em detalhes, mostrando-se os seus conceitos associados, *workflows*, gabaritos, manuais, etc.
- O capítulo 5 descreve uma arquitetura para construção de Ambientes de Desenvolvimento de Software baseada no uso de componentes de prateleira (COTS). A arquitetura é descrita ao nível de componentes utilizados para construção destes ambientes e como estas componentes interagem entre si. Alguns trabalhos relacionados são descritos e comparados com a arquitetura proposta. Uma avaliação desta arquitetura é feita em comparação aos requisitos necessários a um ADS descritos no capítulo 2.
- O capítulo 6 descreve o protótipo de experimentação implementado neste trabalho, mostrando os componentes utilizados, modelagem de processos, interfaces, itens de dados, alguns códigos de implementação, etc. Uma avaliação do protótipo é realizada com base em uma grade de avaliação proposta por Ambriola et. al. [AMV 97] no artigo da *ACM Transactions on Software Engineering and Methodology* cujo título é "*Assessing Process-Centered Software Engineering Environments*".
- O capítulo 7 conclui o trabalho, descrevendo as contribuições e conclusões obtidas a partir do desenvolvimento do mesmo e propondo alguns trabalhos futuros, tanto no mesmo tema de pesquisa, quanto em temas correlatos a este trabalho.

2 Ambientes de Desenvolvimento de Software

Ao longo dos anos, pesquisadores e desenvolvedores de software perceberam a necessidade de ambientes que fornecessem suporte para o desenvolvimento de software, possibilitando a comunicação e coordenação entre os desenvolvedores e todas as ferramentas utilizadas no desenvolvimento de software e manutenção, de modo a permitir a eficiente produção de software de qualidade [BRO 93]. Estes ambientes começaram a aparecer durante a década de 70, com o nome de "Ambientes de Engenharia de Software" (*Software Engineering Environments* - SEE, ou ADS - Ambientes de Desenvolvimento de Software), com a intenção de integrar técnicas de engenharia de software, métodos e ferramentas [OIN 99]. Estes ambientes provêm dispositivos de interação e troca de informação entre os membros de um grupo de desenvolvimento de software. Iteração e troca de informação são fatores importantes que determinam o sucesso ou a falha de qualquer iniciativa de desenvolvimento [BAN 96].

Durante o processo de desenvolvimento de software muitos tipos diferentes de documentos são gerados; estes são usualmente criados como parte de um grupo de trabalho e muitos projetistas compartilham estes documentos [OIN 99].

Com a evolução dos métodos e processos de desenvolvimento de software, começou a se perceber a possibilidade de integrar capacidades de definição e controle de "processos de software" [FIN 94]. Assim, os ADSs agregaram a sua estrutura um conjunto de dispositivos para o suporte, a definição e a execução de processos de software [GIM 99]. Esta agregação de funcionalidades trouxe um novo conceito associado a este tipo de ambiente, os "Ambientes de Engenharia de Software centrados em processo" (ADS-CP) (do inglês: *Process-centered Software Engineering Environments* - PSEE) [BRO 93] [GAR 94] [VER 97] [MAD 90] [BEN 94]. Estes ambientes envolvem o conceito de programação de processos, possuindo como principal característica a descrição e a execução de processos de software, controlando todas as atividades relacionadas aos processos de desenvolvimento de software [GIM 99].

Resumindo, ADS ajudam os desenvolvedores a utilizarem corretamente e consistentemente técnicas de engenharia de software. Estes ambientes também fornecem suporte à integração de ferramentas, o reuso em alto nível de artefatos de software, o uso de uma variedade de métodos, notações e plataformas, a integração de abordagens formais e informais para desenvolvimento de software e o uso de diferentes formalismos de representação para a descrição de processos de software e suas adaptações de acordo com o domínio de aplicação [OIN 99].

Um ambiente ADS-CP é composto por três partes principais [BAN 96] [FUG 96]:

- Um *ambiente de execução (enactment)* ou *motor de processo* (*process engine*) para a execução propriamente dita do modelo de processo. Este deve estar habilitado a invocar ferramentas e recuperar, manipular e armazenar artefatos do processo;

- Um *ambiente de interação com o usuário* ou *Interface com o Usuário* (composto por ferramentas como compiladores, editores diagramáticos e *shells* de controle e configuração) para o suporte ao trabalho do usuário e sua interação com o ambiente;
- Um *repositório de dados* para o armazenamento dos artefatos gerados durante o desenvolvimento e para os modelos de processo.

Em geral, as funcionalidades oferecidas pelos ADS-CPs existentes podem ser resumidas como segue [FUG 96] [ARA 99] [GAR 94] [BAN 96]:

- Estes ambientes são baseados em uma Linguagem de Modelagem de Processos (PML - *Process Modeling Language*) que é usada para criar o modelo de processo. Estas linguagens são baseadas em algum formalismo e possuem uma semântica executável, sendo possível que o modelo de processo seja analisado e interpretado;
- ADS-CPs são integrados com facilidades para gerenciar e armazenar artefatos de processo, isto é, possuem algum dispositivo que funciona como um repositório;
- A ativação de ferramentas de engenharia de software, tais como compiladores, *workbenchs*[*] de análise e projeto, e ferramentas de gerenciamento de configuração, podem ser diretamente modeladas utilizando a PML. Assim, estas ferramentas podem ser diretamente ativadas e controladas pelo interpretador da PML.

Um ADS-CP se baseia na descrição explícita de processo - geralmente definida utilizando uma Linguagem de Definição de Processos (PML - *Process Modeling Language*) - que é construída de acordo com o modelo de processo. Assim, temos uma descrição do processo, que descreve um modelo de processo, que, por sua vez, representa um processo de software [BAN 96] [GIM 99].

As linguagens de modelagem de processos oferecem poderosas capacidades para descrever papéis, procedimentos manuais ou automáticos, interação entre usuários, artefatos do processo e restrições. A execução do modelo de processo dentro de um ADS-CP fornece suporte aos usuários na execução de seu trabalho, por exemplo, guiando-os ou automatizando algumas partes do trabalho [BAN 96].

Um componente importante em um ADS-CP é o "Motor de Processo" (*Process Engine*) que pode ser carregado com o modelo de processo - que especifica e descreve como as atividades de desenvolvimento serão executadas, os papéis e atividades de cada membro da equipe de desenvolvimento e como utilizar e controlar ferramentas de suporte ao desenvolvimento [AMV 97] - da organização que utiliza o ADS-CP. De acordo com o que é descrito no modelo de processo, o motor de processo fornece o ambiente de trabalho apropriado para cada usuário [FIN 94]. Este ambiente adaptado para cada usuário pode ser configurado de acordo com os diferentes papéis que diferentes atores (atores são os usuários do sistema, são os desenvolvedores que trabalham no projeto de um software) podem desempenhar no processo de desenvolvimento, fornecendo as ferramentas adequadas, e meios de ativação automática das mesmas, de acordo com a atividade a ser realizada. O motor de processo executa o

[*] *Workbenchs* são ferramentas que suportam atividades relacionadas a apenas uma das fases do ciclo de vida do software; como análise, projeto, teste, etc.

modelo de processo, levando em conta os eventos que ocorrem no contexto do usuário e as condições definidas para a ativação de novas atividades, fazendo assim a sincronização da execução das atividades, e também deve estar habilitado a ativar ferramentas e recuperar, manipular e armazenar artefatos do processo [FUG 96].

ADS-CPs são uma classe de sistemas que integram as pessoas em uma organização com o processo em execução e a tecnologia utilizada (compiladores, editores diagramáticos, ferramentas de teste, etc.) para o desenvolvimento de produtos de software [GIM 99]. Integração de ferramentas contribui para diminuir o esforço dispendido para conciliar os diferentes requisitos de dados ou métodos usados na construção de software [GIM 94].

ADS-CP e **SGW** (Sistemas de Gerência de Workflow são sistemas para a automação de processos de negócio. Estes sistemas são descritos no Capítulo 3) são tecnologias que lidam com os mesmos tópicos básicos, isto é, como suportar atividades cooperativas em processos centrados em humanos. As principais diferenças, em termos de metas e objetivos, são principalmente relacionadas à terminologia diferente e "conceitos básicos" (*background*) dos domínios onde estas tecnologias se originaram. Enquanto ADS-CP está relacionado a desenvolvimento de software, SGW se aplica a Sistemas de Informação e Processos de Negócio [BAN 96].

2.1 Repositórios de ADS

Durante o processo de desenvolvimento de software muitos tipos diferentes de documentos são gerados; estes são usualmente criados como parte de um grupo de trabalho e muitos projetistas compartilham os mesmos [OIN 99].

Alguns trabalhos enfatizaram o uso de repositórios dentro de ambientes de engenharia de software e sistemas de informação [NOT 2000] [OIN 99] [FUG 96] [BAN 96] [JAR 94].

Um repositório armazena e gerencia todos os artefatos produzidos e manipulados durante o desenvolvimento de software gerenciado pelo ADS, tornando possível o acesso a estes artefatos para uma grande variedade de ferramentas integradas. Um repositório de software é basicamente um banco de dados de artefatos de software; este pode ser utilizado como uma “caixa de depósito segura” para se inserir e retirar modelos com dados de projetos válidos, como uma base para todos os serviços de gerência de dados durante o desenvolvimento do sistema, e como um ambiente comum para atividades em grupo. Repositórios devem ser extensíveis para permitir a sua integração com ferramentas de gerência e com pacotes comerciais existentes como banco de dados padronizados e ferramentas de workflow [OIN 99] [BAN 96]. Um repositório pode ser um banco de dados dedicado (como um banco de dados orientado a objetos), o sistema de arquivos do próprio *host* (computador) do ADS ou mesmo a combinação de um banco de dados e o sistema de arquivos [BAN 96].

Uma maneira de armazenamento dos artefatos resultantes do processo de desenvolvimento é a utilização de documentos XML [XML 2001], possibilitando a integração dos dados em um único meio de armazenamento [BOO 99] [CON 2000], isto é, as diversas ferramentas dentro de um ADS salvariam seus arquivos no formato XML.

A utilização de documentos XML como repositórios de dados em ADS necessita da definição de **DTD´s** (*Document Type Definition*) ou **XML *Schemas*** que definam os modelos de documentos a serem gerenciados pelo ambiente, permitindo o armazenamento de dados oriundos de diferentes ferramentas. XML é uma linguagem mais flexível do que HTML, permitindo que os usuários definan suas próprias *tags*. Isto significa que XML torna possível a adaptação dos documentos para as diferentes necessidades e a representação de diferentes tipos de dados. Para que diferentes ferramentas possam trocar dados entre si - como, por exemplo: um editor de diagramas de classes poder utilizar diagramas de casos de uso para a construção dos modelos de classes e objetos – é necessário que estas ferramentas possibilitem a geração de arquivos em versões XML e que utilizem os mesmos modelos de tipos de documentos (DTD's ou XML *Schemas*) para a geração destes documentos.

Outro mecanismo utilizado em repositórios de dados em ambientes de desenvolvimento são os chamados Dicionários de Dados [NOT 2000] [OIN 99] [JAR 94]. Já que as diversas ferramentas utilizadas dentro de ADS geram diferentes tipos de documentos, a utilização de dicionário de dados possibilita o armazenamento de dados oriundos de diferentes fontes (ferramentas) em um único recurso, o repositório. Este dicionário de dados mantém dois repositórios: (1) um repositório de metadados, que armazena os dados do próprio dicionário, definindo como são disponibilizados os dados vindos das diversas ferramentas utilizadas no desenvolvimento de software e como estes dados serão armazenados no repositório (metadados), e (2) um repositório de dados, que mantém os dados manuseados pelas ferramentas de desenvolvimento do ambiente, que são armazenados de acordo com os seus metadados. Estes repositórios podem ser armazenados em um Sistema de Banco de Dados [NOT 2000].

2.2 Funcionalidades de Hipertexto e Hiperdocumentos nos ADS

Hipertexto (que pode ser representado por grafos direcionados) é um mecanismo de estruturação textual que divide os dados em porções (nodos) relacionados entre si (através de links). Cada nodo contém informações sobre um assunto específico e fornece meios de se atingir (navegar para) outros nodos que contenham informações relacionadas a uma âncora no nodo origem [LIM 2000]. Os benefícios de se incluir funcionalidades de hipertexto em sistemas de aplicação têm se tornado grandemente aceito. Usuários esperam que aplicações possuam funcionalidades de ligação entre documentos (*linking*) e de navegação de documentos (*browsing*) para aumentar as capacidades de procura e de execução de comandos [VER 99].

Ortigosa e Perin [ORT 95] [PER 92] apresentam alguns problemas ou necessidades associadas aos ambientes de desenvolvimento de software (ADS):

- Utilização de diferentes classes de documentos em cada fase do processo de desenvolvimento;
- Os documentos produzidos por uma ferramenta devem utilizados por outras ferramentas em processos posteriores;
- Deve se garantir que todos os documentos sejam coerentes e consistentes durante todo ciclo de vida do software;

- Os documentos descrevem diferentes facetas da mesma realidade. Portanto, devem ser fornecidos mecanismos de alto nível para poder navegar de um documento para outro;
- Inter-relacionamento, e visualização deste inter-relacionamento, das informações contidas nos diversos documentos e artefatos decorrentes dos processos;
- Formulação de caminhos alternativos de visualização (leitura) e manipulação do conjunto de documentos gerados para um software.

Uma técnica importante para a resolução das questões citadas acima é a utilização de hipertexto como mecanismo de representação do conhecimento armazenado durante o processo de desenvolvimento de software [ORT 95] [PER 92].

Hipertexto tem sido utilizado para representar documentos de projeto de software e sistemas; e vários ambientes de engenharia de software baseados em hipertexto estão sendo desenvolvidos [OIN 99].

Existem duas maneiras de utilizar hipertexto no contexto de ambientes de engenharia de software e repositórios [OIN 99]:

- Construir o ADS em volta de hipertexto - neste caso nodos e links são as principais abstrações na aplicação;
- Usar hipertexto como uma funcionalidade que adiciona valor para enriquecer as características do ambiente - nesta abordagem nodos e/ou links são ligados aos objetos da aplicação para fornecer facilidades de navegação, sem qualquer conflito com a funcionalidade existente do domínio que é desvinculada do componente de hipertexto.

Hiperdocumento se refere a documentos com estrutura hipertextual, isto é, que possuem as mesmas características funcionais de hipertexto, mas que possuem informação em formatos diferentes além de texto, como imagens, sons, vídeos, etc.

Uma maneira de fornecer características hipertextuais para um ADS é a construção destes ambientes de modo que os mesmos funcionem com base na plataforma *Internet*. Um ADS pode tirar proveito das características hipertextuais inerentes à *Internet* sendo construído sobre esta plataforma.

A dominância dos navegadores (navegadores) WWW (*World Wide Web*) como interface preferida para um incrível número de tarefas indica a grande aceitação dos benefícios de funcionalidades de hipertexto para aplicações orientadas à tarefas. As capacidades de navegação e ligação de documentos associada com navegação hipertexto são esperadas como parte da interface de qualquer sistema de informação [VER 99].

2.3 Modelagem e Execução de Processos em ADS-CP

Como já mencionado, um ADS-CP é centrado em torno de uma descrição de um modelo de processo, e utiliza para descrever este modelo uma linguagem de modelagem de processos (*PML*).

Estas linguagens normalmente oferecem possibilidades de descrever papéis, procedimentos manuais e automatizados, interação entre usuários, artefatos de processo e restrições. A execução (*enactment*) do modelo de processo dentro do ADS-CP fornece suporte aos agentes do processos (desenvolvedores) na realização de suas tarefas, por exemplo, fornecendo orientações sobre o trabalho ou automatizando alguma parte do processo [BAN 96].

As PMLs existentes são baseadas em uma variedade de diferentes paradigmas e abordagens [FUG 96]:

- O paradigma de Inteligência Artificial: neste modelo de PML o modelo de processo é construído como uma coleção de fatos e regras (como na linguagem Prolog). Fatos podem ser usados, por exemplo, para descrever o estado de artefatos de processos, relacionamentos entre recursos (ex.: um documento e o seu autor) ou restrições de consistência. Regras descrevem como manipular este conhecimento quando condições epecificas são atingidas (ex.: um novo módulo fonte é salvo e por conseqüência pode ser compilado). Um ADS-CP nesta categoria é o Merlin [FIN 94].
- O paradigma de Grafos e Rede: SPADE [BAN 96] utiliza redes de Petri de alto nível estendidas para melhor suportar a modelagem e gerência de processos. A idéia básica é utilizar a capacidade das Redes de Petri de modelarem paralelismo e operações concorrentes. *Tokens* são utilizados para representar recursos e artefatos do processo e regras são associadas com transições para especificar condições e ações associadas.
- O paradigma de Programação de Processos: neste paradigma um processo é descrito utilizando um estilo procedural. O projeto Arcadia [ARC 2001] em seus trabalhos iniciais utilizou uma variante da linguagem ADA chamada APPL/A.
- O paradigma Algébrico/ Funcional: considera processos de software como uma coleção de funções matemáticas as quais mapeiam as entradas de processos em suas saídas.
- O paradigma de Banco de Dados Ativo: Adele/Tempo [FIN 94] é um exemplo deste paradigma. Este ADS-CP é um sistema de banco de dados orientado a objetos modificado com características e mecanismos para suportar modelagem e execução de processos. Adele utiliza os mecanismos de gatilhos (*triggers*) para checar e forçar restrições de consistência.
- O paradigma Híbrido: EPOS [FIN 94] enfatiza o suporte a ambos paradigmas baseado em regras e procedural, e forte acoplamento de técnicas de gerência de configuração e modelagem de processos. EPOS é baseado em um BD Orientado a Objetos (EPOS-DB) e inclui uma linguagem para expressar modelos de processo (SPELL) a qual integra conceitos de orientação a objetos e inteligência artificial.

Segundo Fuggetta [FUG 96], algumas vantagens relacionadas aos paradigmas de linguagens de modelagem de processos apresentados acima são as seguintes:

(a) PMLs baseadas no paradigma procedural oferecem características propícias para modelagem de grandes processos (*modelling-in-the-large*) e para a exploração de características de transparência de informação e princípios de projeto modular.

(b) Abordagens baseadas em regras são convenientes para descrever metas e restrições regidas pelas atividades de desenvolvimento de software. As PMLs deste paradigma também costuman fornecer capacidade de modificar o modelo de processo, mesmo durante a execução.

(c) O paradigma de BD ativo é conveniente para descrever a estrutura do produto de software e as restrições de consistência relacionadas.

(d) Sistemas de Rede de Petri tornam possível a descrição de paralelismo no processo e os estados das diferentes entidades que estão sendo geridas.

(e) Abordagens híbridas são o primeiro passo em direção a integração das características complementares dos diferentes paradigmas.

O ambiente de execução de um ADS-CP executa um processo baseado na descrição deste processo, isto é, um modelo de processo (que representa a abstração escolhida do processo de software) [GIM 94]. O ambiente de execução interpreta o modelo de processo, podendo operar sobre os artefatos armazenados no repositório e coordenar as operações a serem acompanhadas no processo; pode também invocar operações, afetando a interação do usuário com o ambiente, como, por exemplo: ativando ferramentas, modificando o estado das ferramentas ativas e terminando a execução das mesmas. O modelo de processos é construído utilizando-se alguma linguagem de modelagem de processos (*PML*). As PMLs podem apresentar diferentes paradigmas de modelagem, como mostrado acima.

2.4 Requisitos necessários para um ADS

A produção de software de qualidade requer ambientes de desenvolvimento que assistam o processo de produção, seguindo estritos padrões de controle a respeito da metodologia e os processos utilizados para o desenvolvimento de software [ORT 95].

Um ADS é uma coleção de funcionalidades integradas para auxiliar desenvolvedores e gerentes a realizarem as suas funções [ARC 2001], com intuito de se produzir software de qualidade. Software de qualidade está diretamente ligado à metodologia e aos processos utilizados para sua construção [ORT 95]. Vários pesquisadores e projetos de pesquisa identificaram os requisitos necessários para um ambiente com suporte a processo de software [ARC 2001] [BAR 2000] [CAL 96] [CHA 97] [CHA 97a] [CHA 99] [FIN 94] [GIM 94] [MAN 2001] [MAU 99] [ORT 95] [PER 92] [NOT 2000]. Estes requisitos são descritos a seguir.

- **Suporte ao Gerenciamento**: um ADS deve fornecer visibilidade dos projetos em andamento, suportar as tomadas de decisões e controle do projeto por parte dos gerentes, auxiliar o controle do acesso aos dados e atribuir tarefas para as pessoas engajadas ao projeto [ARC 2001]. As informações referentes a um projeto - como: quais são os responsáveis pelas atividades, quais são os artefatos associados a uma atividade, qual o resultado de uma atividade de revisão de um artefato, qual a data de finalização de uma tarefa, etc. - são armazenadas em um repositório do ADS, permitindo ser recuperadas quando necessário.
- **Suporte aos Eventos do Processo**: um ADS deve suportar eventos como notificar o usuário que é designado para executar determinada atividade do processo, disseminar as informações necessárias à execução da atividade, e coletar informações geradas como resultado da execução da atividade pelo usuário [CHA 97]. Esta funcionalidade pode ser obtida através da utilização de um "motor de processo" [BAN 96] [FUG 96], que é responsável pelo monitoramento dos eventos dentro do ADS e pode disparar ações de acordo com os eventos e condições pré-definidas.
- **Flexibilidade**: as necessidades dos desenvolvedores e dos usuários são diversas, variando de projeto para projeto e de tempos em tempos, com diferenças e alterações de cargos. É necessário que o ambiente possibilite alterações dinâmicas do processo [ARC 2001] [CHA 97]. É necessário também que o ambiente possua Adaptabilidade de Metodologias, visando possibilitar a aplicação de diferentes metodologias, que se adaptam à realidade da empresa e ao domínio do problema do software que será construído [GIM 94]. A possibilidade de modificação dos processos pode ser alcançada através da disponibilização de linguagens (gráficas ou textuais) de modelagem de processos que permitem - além da descrição de papéis, de procedimentos manuais e automatizados, de interação entre usuários, de artefatos de processo e restrições [BAN 96] - a modificação dinâmica do processo de um ADS-CP. A adaptação de metodologias é conseguida pela possibilidade de adição de novas ferramentas ao ambiente e pela disponibilização de diferentes informações e gabaritos para a execução das atividades.
- **Extensibilidade**: refere-se à possibilidade de se integrar novas ferramentas ao ambiente à medida que produtos mais avançados surgem [GIM 94] [ARC 2001]. Linguagens de modelagem de processos podem ser utilizadas para especificar quais ferramentas devem ser ativadas para a execução das atividades. Uma alternativa adicional é a disponibilização de interfaces que permitam ao usuário a definição (cadastro) de dados referentes a novas ferramentas. Estes dados (como a pasta que contém o executável referente à ferramenta) podem ser mantidos no repositório do ADS e serem utilizados no momento da ativação das ferramentas.
- **Reusabilidade**: o reuso de objetos em desenvolvimento de software é importante no que tange questões de qualidade e redução de custos e esforços [GIM 99]. Reusabilidade pode ser obtida pela utilização de objetos prontos e que proporcionem funcionalidades específicas dentro de um projeto, como por exemplo: casos de projeto e padrões para as tarefas executadas constantemente (como padrões de projeto e padrões de processo), modelos parametrizados e genéricos para as estruturas e comportamentos que acontecem freqüentemente (como gabaritos e manuais de orientação), generalização e especialização de entidades similares, etc. [CHA 97].

- **Colaboração**: o desenvolvimento de grandes sistemas de software requer a colaboração de diversos usuários, que não estão necessariamente localizados em um mesmo local. Um ADS deve suportar a interação e comunicação efetiva entre os usuários [CHA 97] [BAN 96] [ARC 2001] [CAL 96]. A construção do ADS com base na plataforma *Internet* pode fornecer meios para *Colaboração* entre os desenvolvedores. A *Internet* é inerentemente uma plataforma colaborativa, fornecendo meios de acesso de qualquer ponto do planeta.
- **Fácil de Usar**: se a ferramenta é para apoiar os usuários, ela não deve se tornar um obstáculo. Os usuários devem entender as capacidades que o ADS oferece e como usá-lo efetivamente. Ambientes que requerem muito treinamento ou possuem muitas características díficeis de serem aprendidas pelo usuário tendem a cair no desuso [ARC 2001] [GIM 94].
- **Monitoramento**: monitoramento é crucial nas práticas de gerenciamento atuais. A coleção de dados do sistema, principalmente dados de performance, e sua apresentação apropriada podem prover uma percepção valiosa sobre o processo de software. Esta funcionalidade irá auxiliar no melhoramento do uso efetivo dos recursos que podem ser redistribuídos para alcançar a melhor qualidade de software [CHA 97]. Coletar métricas e seguir um processo de desenvolvimento disciplinado são tarefas difíceis em qualquer projeto de software [CAL 96]. Dados de projeto podem ser armazenados em um repositório, sendo possível recuperá-los quando necessário e serem apresentados aos usuários. O armazenamento de informações, como: início e término de atividades, desenvolvedores assinalados como responsáveis pelas atividades, artefatos produzidos durante a atividade, etc., pode permitir uma análise e produção de relatórios que mostrem o andamento dos projetos e onde se localizam os “gargalos”, isto é, onde que o processo pode ser melhorado. Esta melhora do processo pode ser realizada pela divisão de uma atividade em sub-atividades, pela alocação de um maior número de desenvolvedores para execução das atividades, ou pela utilização de ferramentas mais eficazes, etc..
- **Consistência**: a consistência de dados compartilhados é importante em um ambiente de suporte a múltiplos usuários, onde os dados podem ser acessados de vários locais e em qualquer tempo. Desde que atividades de desenvolvimento de software são tipicamente de longa duração, um bom suporte a transações deve garantir um bom grau de consistência sem ser pessimista demais em relação à disponibilidade dos dados [CHA 97]. A consistência dos dados pode ser obtida através da utilização de sistemas de gerência de banco de dados como o repositório dos dados dentro do ADS. A consistência dos dados pode ser garantida através de propriedades ACID das transações [SIL 99] - Atomicidade (*Atomicity*), Consistência (*Consistency*), Isolamento (*Isolation*) e Durabilidade (*Durability*) – normalmente fornecidas pelos Sistemas de Gerência de Banco de Dados. Os dados oriundos de diferentes ferramentas aparecem com o problema já mencionado na seção 2.1, que é: “*como integrar os dados de diferentes ferramentas?*”. Duas possíveis soluções foram mostradas: (a) utilização de documentos XML e (b) utilização de dicionários de dados.
- **Verificação**: a definição explícita do processo de software permite que o mesmo seja revisto e melhorado [CHA 97]. Uma linguagem de modelagem de processos (preferencialmente uma linguagem gráfica com notações como diagramas de estados ou de atividades [LAR 2000] [FOW 2000] [BOO 2000] [MAR 2000] ou redes de

Petri [BAN 96]) agregada ao ADS pode permitir a verificação dos processos modelados pela disponibilização de dispositivos de simulação ou depuração agregadas à própria linguagem de modelagem de processos.

- **Integração de Controle**: é a habilidade de ativar as várias ferramentas que podem ser usadas no desenvolvimento de um produto de software, e capturar seus resultados, isto é, um meio de compartilhamento de funções entre ferramentas, possibilitando o controle de comportamento destas ferramentas [CHA 97] [BAN 96] [JAR 94]. A linguagem de modelagem de processos pode especificar as diferentes ferramentas que devem ser ativadas para a execução das diferentes atividades definidas no modelo de processo. Outra maneira para fornecer a integração de ferramentas é o armazenamento, através de um repositório, dos dados referentes às ferramentas (como a pasta do sistema de arquivos que contém o executável referente à ferramenta) que estão associadas à execução de determinadas atividades. De posse destes dados, é possível se ativar as devidas ferramentas quando do início da execução de uma nova atividade por parte de um desenvolvedor. Uma observação importante em relação ao cadastro de dados sobre uma determinada ferramenta é que estes dados precisam ser armazenados em um repositório compartilhado e é dependente do local (o computador) de acesso ao ambiente.
- **Integração de Dados**: integração de dados se refere ao problema de combinar dados de diferentes fontes (e possivelmente diferentes formatos de dados), e fornecer ao usuário, ou a um ambiente, uma visão unificada destes dados, como se todos possuíssem o mesmo formato [LEN 2002]; integração de dados trata da habilidade de ferramentas trocarem dados entre si com um pequeno esforço de conversão e um alto grau de consistência [JAR 94]. Uma maneira simplista de integrar os dados em um ADS é o agrupamento em um único documento (hipertexto, hiperdocumento ou um documento XML) todos os artefatos resultantes de uma seqüência de atividades do processo, permitindo o fácil acesso aos mesmos e a representação do conhecimento adquirido durante o processo de desenvolvimento [ORT 95] [PER 92]. As diversas ferramentas utilizadas no processo de desenvolvimento também poderiam compartilhar a informação e os artefatos gerados em um banco de dados comum, compartilhado por todos. Neste contexto surge a necessidade de uma ferramenta de dicionário de dados, como proposto por Notari [NOT 2000] e outros. Este tipo de ferramenta fornece maneiras de armazenar os dados de diferentes ferramentas em um único banco de dados, que é compartilhado por todas ferramentas do ambiente de desenvolvimento (veja seção 2.1). XML também é uma maneira de compartilhar dados, entretanto **DTDs** (*Document Type Definition*) ou **XML *Schemas*** [XML 2001] [BOO 99] [CON 2000] devem ser definidos para permitir o armazenamento de dados de diferentes ferramentas.

2.5 Resumo do Capítulo

Ambientes de Engenharia de Software (ADS) fazem parte de uma classe de ambientes complexos e que possuem várias funcionalidades. A utilização de repositórios dentro destes ambientes, com o intuito de realizar o armazenamento dos dados referentes aos projetos de software é uma característica importante, pois visa além

de armazenar as várias informações e documentos associados aos projetos, facilita a recuperação de informação, visto que a procura pode ser feita em apenas uma fonte de dados, o repositório. Funcionalidades hipertextuais (como as encontradas em páginas *Web*) parecem ser uma alternativa atraente para a representação dos artefatos de projeto e conhecimento adquiridos no decorrer do ciclo de vida dos softwares.

Segundo vários autores, ADS devem possuir algumas características que são fundamentais para o sucesso de utilização deste tipo de ambiente e para a produção de software de qualidade.

Para a construção destes ambientes e obtenção destas funcionalidades, pode se pensar na utilização de ferramentas prontas, disponíveis comercialmente, que possuam interfaces bem definidas e forneçam serviços que desempenhem tais funcionalidades. Um bom exemplo deste tipo de contrução é a aplicação de Sistemas de Gerência de *Workflow* (SGW) como o motor de processo de um ADS-CP. A construção de ambientes de desenvolvimento de software a partir da utilização de ferramentas comerciais é discutida no capítulo 5.

3 Sistemas de Gerência de *Workflow*

A *Workflow Management Coalition* (WFMC) define Sistemas de Gerência de *Workflow* (SGW) como: “a automação de um processo de negócio, em todo ou parte, durante o qual documentos, informações ou tarefas são passadas de um participante para outro para realização de alguma ação, de acordo com um conjunto de regras procedimentais” [WMC 99].

Workflow pode ser descrito como o movimento de documentos e tarefas através de um processo de negócio. *Workflow*'s podem ser seqüências de atividades de trabalho ou mesmo um conjunto complexo de processos, sendo que podem ocorrer processos executados concorrentemente e geralmente causando algum efeito na execução dos demais de acordo com os conjuntos de regras, rotas e papéis relacionados aos mesmos [DIC 97].

Workflow pode ser considerado como o fluxo de informação e controle em um processo de negócio [GAL 2000] ou como uma coleção de *tarefas* organizadas para acompanhar alguns processos de negócio. Um *Workflow* define a ordem das invocações das tarefas ou a(s) condição(ões) sobre a(s) qual(is) a(s) tarefa(s) deve(m) ser invocada(s), a sincronização da(s) tarefa(s) e fluxo de informação (*dataflow*). O conceito de *Workflow* tem envolvido desde a noção de processo em manufatura até em escritório [GEO 95].

A *Ultimus Inc.* ™ [WOR 2000] define *Workflow* como : "*Uma seqüência de tarefas estruturadas ou semi-estruturadas realizadas em série ou em paralelo, por dois ou mais indivíduos, com o intuito de alcançar um objetivo comum*".

No princípio da industrialização, um processo era controlado e "conduzido" inteiramente por humanos, que manipulavam objetos físicos. Quando do aparecimento da Tecnologia de Informação, os processos nos locais de trabalho foram parcialmente, ou mesmo totalmente automatizados por Sistemas de Informação, isto é, programas de computadores executanto, ou ativando, as tarefas e fazendo cumprir as regras que foram implementadas previamente por humanos (analistas, programadores, etc.) [GEO 95].

Uma tarefa pode ser realizada por um ou mais sistemas de software, um ou um grupo de humanos, ou ainda uma combinação de ambos [GEO 95].

Os sistemas de gerência de *Workflow* (SGW) permitem que as organizações **definam** e **controlem** várias atividades associadas aos processos de negócio. Estes sistemas estão inseridos na área de CSCW (*Computer Supported Cooperative Work*) e são sistemas que permitem a definição, criação e gerência de *Workflows* através do uso de software, executado em um ou mais "*Workflows engine*" (motores de *Workflow*).

Estes motores de *Workflow* são capazes de interpretar a definição do processo, interagir com os participantes do *Workflow* e, quando necessário, invocar ferramentas e aplicações de sistemas de informação. *Workflow* pode ser uma progressão seqüencial de atividades de trabalho ou um complexo conjunto de processos, cada um sendo executado concorrentemente e causando impacto sobre os demais de acordo com os conjuntos de regras, procedimentos e papéis (funções) [DIC 97] [WMC 99].

Os objetivos de um SGW são em primeira fase a modelagem da estrutura de uma empresa e da seqüência de todos os procedimentos de negócios, e em uma segunda fase o controle, supervisão e registro da execução de todos os processos modelados. Uma seqüência de procedimentos de negócio é modelada em um *processo de negócio*. Todos relacionamentos seqüenciais e paralelos entre os passos de processo são especificados em modelos de processo de negócio. Este determina para cada passo quais são os objetos de trabalho (dados e documentos) e quais recursos humanos, técnicos e organizacionais que são necessários para execução de determinado passo do processo. Um processo de negócio é modelado como um grafo, com os passos do processo como nodos e os relacionamentos de controle e fluxo dos dados como arestas. Um modelo de processo executável é denominado *Workflow* [SCH 97].

Processos de *Workflow* são, freqüentemente, coleções distribuídas de atividades que envolvem grupos de indivíduos em localizações diferentes. Para coordenar estas tarefas, um sistema de suporte a processo deveria fornecer para processos distribuídos a execução e integração com ferramentas através da rede [HIT 98].

A aplicabilidade de *Workflow* é muito vasta, visto que a maioria das atividades dentro de empresas, organizações, universidades, etc. são realizadas de uma maneira padrão, seguindo certos protocolos e processos que são definidos previamente e que devem ser seguidos. Estes processos geralmente possuem documentos ou produtos associados, que por sua vez são, geralmente, de fato o "resultado" obtido pela realização destes processos. Desta maneira, pode-se aplicar sistemas de *Workflow* para o controle dos processos, por exemplo, desde sistemas de venda e reservas de passagens aéreas até em ambientes de desenvolvimento de software, realizando o controle das atividades realizadas dentro do ambiente.

3.1 Padrões da WFMC para Sistemas de Gerência de *Workflow*

A *Workflow Management Coalition* (WFMC) [WMC 2001] é uma organização internacional e sem fins lucrativos fundada em Agosto de 1993 por vendedores de produtos de *Workflow*, usuários, analistas, grupos de pesquisa e universidades. Seus objetivos principais são promover e desenvolver o uso de *Workflow* através do estabelecimento de padrões para a terminologia de software, interoperabilidades e conectividades entre produtos de *Workflow*.

Uma das contribuições da WFMC foi a definição de um modelo de referência para os Sistemas de Gerência de *Workflow* (*SGW*). Este modelo oferece um *framework* arquitetural do trabalho dos SGW's. Este modelo possui 5 interfaces, as quais cobrem as áreas de funcionalidade entre os SGWs e os seus ambientes. As áreas de funcionalidades são as seguintes [WMC 99] [HOL 95]:

1. A importação e exportação de definições de processo;
2. Interação com aplicações clientes e software de manuseio de listas de serviços (*worklists*);
3. A invocação de ferramentas e aplicações de software;

4. Interoperabilidade entre diferentes SGW's;
5. Funções de Administração e Monitoramento.

O modelo de referência referido acima pode ser visualizado graficamente na Figura 3.1.

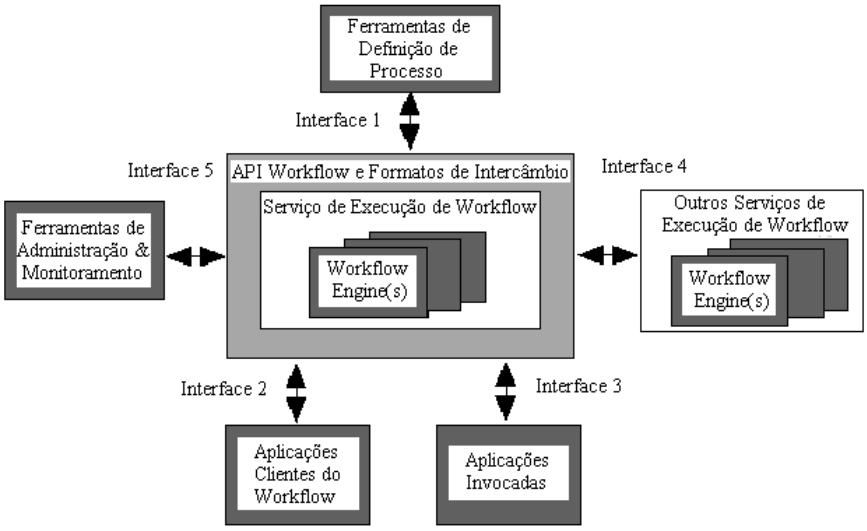

FIGURA 3.1 - Modelo de Referência de SGW [HOL 95]

O modelo de referência introduz a abstração de "Serviço de Execução de *Workflow*" (*Workflow-enactment service*) para representar qualquer tipo de ambiente de execução para uma aplicação componente de controle de fluxo de *Workflow*. O componente de controle de fluxo pode ser realizado por objetos de negócio "vivendo" em um servidor de objetos de negócio ou por um modelo de processo interpretado por um motor de *Workflow* (*Workflow engine*). A partes internas do serviço de execução de *Workflow* estão fora do escopo de padronização da WFMC.

O modelo de referência identifica as áreas principais de interação entre componentes desenvolvidos independentemente de uma variedade de soluções de *Workflow*. A seguir são listados os tipos de padrões identificados no modelo de referência [SCH 99]:

- **Padrão de Intercâmbio de Definição de Processo** - define como as ferramentas de modelagem de processos de *Workflow* e o serviço de execução de *Workflow* devem trocar definições de processo. Algumas realizações incluem a meta-linguagem de definição de processo de *Workflow* e a linguagem WPDL (*Workflow Process Definition Language*), e o padrão OMG XMI (*XML Meta Data Interchange*) e a linguagem UML (*Unified Modeling Language*).
- **Padrão de Interação Cliente-Aplicação** - permite que os participantes interajam com o *Workflow* para controlar a execução do processo e manipular itens de trabalho de uma maneira unificada.
- **Padrão de Interação Aplicação-Componente** - define a interação entre um componente de controle de fluxo de uma aplicação *Workflow* e os componentes da aplicação. Estes padrões suportam a geração de adaptadores para as aplicações existentes e fornecimento de "linhas guias" (*guidelines*) para o desenvolvimento de novos componentes de aplicações de *Workflow*.

- **Padrão de Interoperabilidade entre Aplicação-*Workflow*** - permite independentemente desenvolver e gerenciar a interação entre aplicações de *Workflow* para a realização de processos de *Workflow* complexos.
- **Padrões de Administração e Monitoramento** - habilita a administração consistente de diferentes aplicações de *Workflow* e permite que os eventos de mudanças de estado sejam comparados.

Uma outra contribuição interessante da WFMC foi a Estrutura Genérica de Produtos de *Workflow* [WMC 99]. Esta estrutura melhora a compreensão da funcionalidade de SGW e pode fornecer uma base comum para a construção de produtos de *Workflow* e desenvolvimento de cenários de interoperabilidade [HOL 95]. A estrutura genérica de produtos de *Workflow* pode ser visualizada na Figura 3.2.

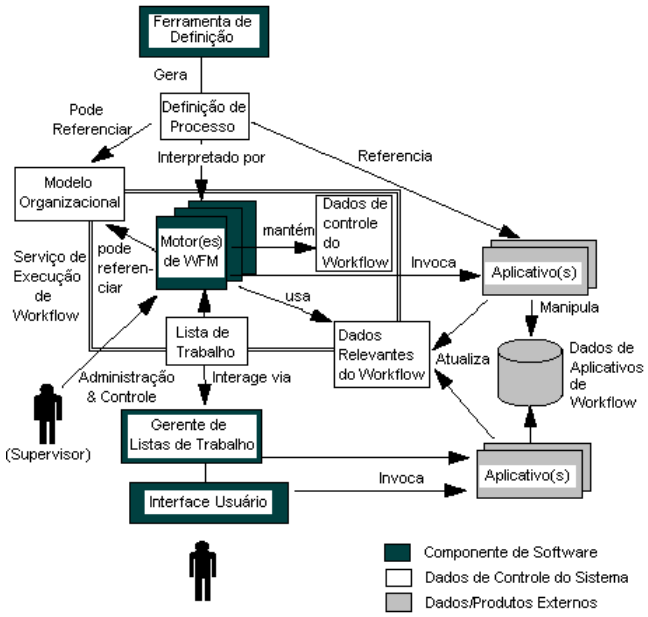

FIGURA 3.2 - Estrutura Genérica de Produtos de *Workflow* [HOL 95]

A estrutura genérica identifica os principais componentes de um sistema de *Workflow* mais as respectivas interfaces, provendo um modelo abstrato. Este modelo pode ter muitas implementações, diferindo em aspectos como plataformas, protocolos de rede e tecnologia de distribuição. Conseqüentemente, as interfaces especificadas devem suportar esta diversidade de ambientes operacionais. Os principais componentes citados neste parágrafo são descritos a seguir:

- **Ferramenta de Definição** (*Definition Tool*): esta é utilizada para transformar uma definição de processo em um programa executável. Esta ferramenta pode ser baseada em uma linguagem de definição de processo formal, um modelo de relacionamentos de objetos, ou, em sistemas mais simples, um *script* ou um conjunto de comandos de roteamento para transferir informação entre os usuários [HOL 95].
- **Definição de Processo** (*Process Definition*): a definição de processo contém toda a informação necessária sobre o processo para habilitar a execução do mesmo por parte do software de execução de *Workflow*. A definição de processo contém informações sobre as condições de início e fim, atividades constituintes e as regras de navegação entre as mesmas, tarefas de usuarios a serem realizadas, referencias às

aplicações que podem ser invocadas, definição de qualquer dado relevante do *Workflow* que possa ser referenciado, etc. [HOL 95].

- **Serviço de Execução de *Workflow*** (*Workflow Enactment Service*): este é o "coração" de um sistema de *Workflow*. Este interpreta a descrição do processo e controla a instanciação dos processos e a seqüência de atividades, adicionando itens de trabalho para as listas de trabalho dos usuários e invocando os aplicativos quando necessários. Dentro deste serviço existe o "motor de *Workflow*" (*Workflow engine*), os dados relevantes do *Workflow* e os dados de controle do *Workflow*. O "Motor de *Workflow*" é o modulo de software que efetivamente controla o processo, gerenciando suas respectivas instâncias. Os dados relevantes do *Workflow* são os dados produzidos ou atualizados pelos aplicativos, mas que são acessíveis pelo motor de *Workflow*, e são utilizados os controles do *Workflow*. Os dados de aplicativos são manipulados pelos aplicativos de *Workflow* [HOL 95].
- **Listas de Trabalho** (*Worklists*): o motor de *Workflow* aloca itens nas listas de trabalho para que o Gerenciador de Listas de Trabalho tome as ações necessárias. O gerenciador de listas de trabalho irá gerenciar as interações com os participantes do *Workflow*. O assinalamento de itens de trabalho para os participantes pode ser feito estaticamente (durante a definição do processo) ou dinamicamente [HOL 95].
- **Gerenciador de Listas de Trabalho e Interface Usuário** (*Worklists Handler/User Interface*): os participantes interagem com suas respectivas listas de trabalho através de um Gerenciador de Listas de Trabalho. Este componente representa a interface entre o usuário e o Serviço de Execução de *Workflow*. Através desta interface, os itens de trabalho são apresentados ao usuário, permitindo que o usuário informe o término de suas atividades [HOL 95].
- **Aplicativos** (*Applications*): são ferramentas externas necessárias para a realização das atividades. Os aplicativos pode ser editores de texto, planilhas eletrônicas, formulários eletrônicos, etc. [HOL 95].
- **Supervisão** (*Supervision - Administration & Monitoring*): em um sistema de *Workflow* existem algumas funções de supervisão, como alterar regras de alocação de tarefas. Para executar essas funções é necessário privilégio de supervisão, que normalmente é fornecido para uma estação de trabalho ou para um usuário particular [HOL 95].

3.2 Groupware

Sistemas de *Workflow* estão inseridos na área de *Groupware*, sendo que esta se refere a qualquer produto de software ou tecnologia que habilita grupos de pessoas a trabalharem juntas, cooperando entre si.

A **cooperação** se dá entre agentes (software ou humanos) que trabalham juntos para alcançar uma meta comum dentro de uma organização. Cooperação é composta sobre quatro elementos: (1) Colaboração é a criação e gerência de informação compartilhada; (2) Comunicação é a troca de conhecimento e informação entre agentes; (3) Coordenação é a gerência e o ajuste dos esforços de um grupo ou organização de

agentes; produtos de *workflow* se preocupam com este conceito; e (4) Acoplamento (*Coupling*) é o uso de visões compatíveis das informações compartilhadas [OCA 98].

Existem três categorias de produtos de *Groupware*: de ***Comunicação***, de ***Colaboração*** e de ***Coordenação*** [WOR 2000].

Os produtos de ***Comunicação*** são aqueles que permitem que os usuários se comuniquem rápida e facilmente uns com os outros. A comunicação em grupos de trabalho segue alguns atributos que devem ser previstos pelos produtos nesta categoria [WOR 2000]:

1. Estes são em sua maioria *ad hoc* ou aleatórios; não existe nenhuma estrutura ou processo;
2. Deve ser rápido e fácil de usar;
3. Deve ser de baixo custo;
4. Deve ser tão grandemente empregado quanto possível.

Exemplos deste tipo de produto incluem E-mail, fax, telefonia por computador, vídeo conferencia e programas de *chat*.

Produtos de ***Colaboração*** envolvem trabalhadores (referenciados como *knowledge workers*) que trabalham juntos como um time em projetos como produção de relatórios, projetos de produtos complexos ou participação em pesquisas. Soluções de *Groupware* colaborativas devem atender às necessidades de indivíduos trabalhando em projetos conjuntos [WOR 2000]:

1. Estas soluções devem fornecer um "documento" ou repositório onde o trabalho coletivo do grupo é armazenado e facilmente acessado por todos os participantes;
2. Devem fornecer meios de restrições de acesso aos documentos para ver *quem* tem direito de fazer *o quê*;
3. Deve ser fácil de utilizar e não-intruso. Por outro lado, isto irá impedir a criatividade, a qual é essencial para os trabalhadores;
4. Eles devem concordar com as atividades de outros trabalhadores.

Alguns exemplos de produtos deste tipo são Lotus Notes, sistemas de gerência de documentos, software de projeto gráfico, CAD e outras aplicações multi usuários.

Para complementar as características de comunicação e colaboração, existem produtos que permitem que indivíduos também trabalhem juntos participando de processos estruturados ou semi-estruturados. Estes são produtos de *Groupware* classificados como de ***coordenação*** e que também são conhecidos como ***Workflow***. Estes tipos de produto devem ter as seguintes características [WOR 2000]:

1. O "processo" é a essência do *Workflow* ou *coordenação*. A solução deve, portanto, habilitar uma organização para implementar efetivamente seus processos de negócio;
2. Processos podem ser estruturados ou semi-estruturados. Não devem ser puramente *ad hoc*;

3. Coordenação é "pro-ativa". Isto tem o propósito de se obter as metas ou saídas desejadas;
4. Cada organização tem um grande número de processos de negócio. Coordenação (ou *Workflow*) é predominante em cada organização de alguma maneira.

Exemplos de *Workflow* incluem ordens de compra, processamento de reclamações, revisões, relatórios de gastos, procedimentos de produção, suporte para clientes, etc.

3.3 Tipos de *Workflow*

Segundo Dicaterino et. al. [DIC 97], aplicações de *Workflow* são geralmente divididas em 4 categorias, diferenciadas principalmente pelo mecanismo de transporte utilizado para rotear os itens de trabalho. As categorias são as seguintes:

1. *Sistemas de Workflow de Produção* - estes sistemas fazem o roteamento de um ou mais formulários ou diferentes tipos de documentos através da organização. Tipicamente este tipo de *Workflow* armazena os documentos em um repositório central e fornece checagem de entrada do documento (*check-in*), checagem de saída do documento (*check-out*) e controle de versão para estes documentos. Estes sistemas têm como vantagem o suporte de mais características e funções dos que ferramentas baseadas em mensagens, permitem grande adaptação e “rodam” em um grande campo de ambientes de rede e computação. Como desvantagem destes sistemas, pode-se citar o custo mais elevado.
2. *Sistemas de Workflow baseado em Mensagens* - algumas vezes estes também são chamados de *sistemas de Workflow administrativo*. Os produtos contidos nesta categoria são ferramentas *stand-alone* que fazem o roteamento de documentos através dos sistemas de correio eletrônico existentes. São produtos geralmente de baixo custo e suportam a rápida definição e ativação de processos de negócio simples. Estes sistemas não são tão flexíveis quanto os sistemas de *Workflow* de produção.
3. *Sistemas de Workflow baseado na Web* - estes sistemas utilizam clientes e servidores Web para alcançar as suas funcionalidades. Estes sistemas oferecem a vantagem de ser implementados em cima de uma tecnologia que já existe nos ambientes das organizações, a Web. Estes sistemas necessitam de um alto grau de habilidade para desenvolvimento e emprego do sistema, pois não é esperado que os usuários finais desenvolvam formulários ou aplicações Java.
4. *Sistemas de Workflow baseados em Conjunto de Ferramentas Integradas (Suite-based)* - os produtos desta categoria oferecem uma série integrada de aplicações de escritório tais como processadores de texto, planilhas, apresentações e mail eletrônico. Nesta categoria de sistemas de *Workflow*, todas as aplicações são de alguma forma integradas com o sistema de correio eletrônico. Esta integração é freqüentemente aperfeiçoada através de comandos de “*send*” e de “*add routing slip*” na estrutura de menu das aplicações de correio não eletrônicas.

Não existe "a melhor categoria" de sistemas de *Workflow*. O sistema certo depende da natureza dos processos a serem suportados pela ferramenta de *Workflow*. Diferentes ferramentas suportam níveis variantes de estrutura. Enquanto sistemas de *Workflow* baseados em Ferramentas Integradas (*suite-based* - conjunto de ferramentas integradas no intuito de formar um sistema completo) devem suportar compartilhamento de dados interpessoal de maneira *ad-hoc*, sistemas de *Workflow* de produção são melhores no suporte de processos de trabalho rígidos e bem definidos. As demais categorias podem ser representadas entre estas duas categorias [DIC 97].

3.4 Características Típicas dos Sistemas de Gerência de *Workflow*

A maioria dos SGW possuem algumas características típicas, as quais são listadas e descritas a seguir [DIC 97] [WOR 2000]:

- **Ferramenta de Definição de Processo -** uma ferramenta gráfica ou textual para definição dos processos de negócio. Dentro de um processo, cada atividade está associada a uma pessoa ou a uma aplicação de computador. As regras são criadas para determinar como as atividades progridem através do *Workflow* e quais controles governam cada atividade. Existem alguns sistemas que permitem mudanças dinâmicas para o processo de negócio selecionando pessoas com poder (*clearance*) administrativo [DIC 97].
- **Simulação, Prototipação e Versões "Piloto" (*Piloting*) -** alguns sistemas permitem simulação de *Workflow* ou criação de protótipos e/ou versões piloto de um *Workflow* em particular que é experimentado e testado em uma configuração limitada antes do mesmo ir para produção [DIC 97].
- **Iniciação e Controle de Tarefas -** o processo de negócio definido é iniciado e os recursos tecnológicos (IT) e humanos apropriados são agendados e/ou engajados para completar cada atividade do progresso do processo [DIC 97].
- **Regras baseadas no Processo de Decisão -** regras são criadas para cada passo com o intuito de determinar como os dados relacionados com o *Workflow* são processados, roteados, monitorados e controlados. Por exemplo, uma regra deve gerar notificações de correio eletrônico quando uma condição for alcançada. Outra regra deve implementar roteamento condicional de documentos e tarefas baseada no conteúdo dos campos. Alguma outra deve invocar uma aplicação em particular para visualizar dados [DIC 97].
- **Roteamento de Documentos -** em sistemas simples, isto deveria ser aperfeiçoado pela passagem de um arquivo ou pasta de um participante (*recipient*) para outro (por exemplo, uma mensagem de correio eletrônico com anexos). Em sistemas mais complexos, isto deveria ser acompanhado pela checagem de documentos fora de um repositório central. Ambos sistemas devem permitir a edição/revisão com controle de alterações dos documentos (*redlining*), de modo que cada pessoa no processo pode adicionar seus próprios comentários sem afetar o documento original [DIC 97].

- **Chamada de Aplicações para Visualizar e Manipular Dados -** processadores de texto, planilhas, sistemas de informações geográficas, aplicações de produção, etc. podem ser chamados para permitir aos trabalhadores criar, atualizar e visualizar dados e documentos [DIC 97].
- **Listas de Trabalhos/Tarefas (*Worklists*) -** estes permitem que cada trabalhador rapidamente identifique suas tarefas atuais (*current tasks*) juntamente com prazos, prioridades, etc. Em alguns sistemas, a carga de trabalho antecipada (*antecipated workload*) pode ser mostrada também. Estes sistemas analisam onde os trabalhos estão no *Workflow* e quanto tempo cada passo vai demorar, e então faz estimativas quando várias tarefas irão alcançar a mesa de um indivíduo [DIC 97].
- **Automação de Tarefa -** Tarefas computadorizadas podem ser automaticamente invocadas. Isto deve incluir coisas como escrita de cartas, notificações de email, ou execução de aplicações de produção. Automação de tarefas geralmente requer a customização de produtos de *Workflow* básicos [DIC 97].
- **Notificação de Eventos -** empregados e/ou gerentes podem ser notificados quando certos eventos ocorrem, quando a carga de trabalho aumenta, etc. [DIC 97].
- **Listas de Distribuição (*Roteamento*) para Mensagens/Mail -** listas de distribuição podem ser criadas para o envio de mensagens *ad-hoc* entre os empregados [DIC 97].
- **Monitoramento de Processo -** o sistema pode prover informação valiosa sobre a carga de trabalho atual, futuras cargas de trabalho, gargalos (atuais ou em potencial), (*turn-around time*) tempos médios, prazos não cumpridos, etc. [DIC 97].
- **Acesso a Informação através da *World Wide Web* -** alguns sistemas fornecem módulos de interface para fornecer informação do *Workflow* para clientes remotos, fornecedores, colaboradores, ou empregados [DIC 97].
- ***Tracking* e *Logging* das Atividades -** informação sobre cada passo pode ser armazenada (***log***). Isto deve incluir coisas como tempos de início e fim das atividades, pessoa(s) assinalada(s) para a tarefa, e o campo de *status* da tarefa. Esta informação deve mais tarde usada para análise do processo ou fornecer evidência que certas tarefas de fato foram completadas [DIC 97].
- **Administração e Segurança -** um número de funções são usualmente fornecidas para identificar os participantes e seus respectivos privilégios bem como para administrar rotinas associadas com qualquer aplicação (ex., arquivos de *back-ups*, arquivos de *logs*) [DIC 97].
- **Papéis (*roles*) -** habilidade de assinalar tarefas para "papéis (*roles)*" ou "funções de trabalho (*job functions*)" para que o projeto do *Workflow* não necessite ter que ser modificado cada vez que um usuário muda uma função ou responsabilidade [WOR 2000].
- **Regras (*rules*) -** possibilidade de integrar a lógica de um processo de negócio complexo na definição do *Workflow* sem a necessidade de programação ou *script* [WOR 2000].
- **Manuseio de Exceção (*Exception handling*) -** habilidade de manusear exceções, as quais estão sempre presentes em qualquer organização. Por exemplo, a habilidade de reassinalar uma tarefa crítica de um usuário para outro se o usuário está ausente por

causa de uma emergência e o seu computador está com proteção de tela [WOR 2000].

- **Pró-Ativo (*Pro-Active*) -** o *Workflow* funciona de uma maneira pro-ativa. A solução de *Workflow* deve informar os usuários de novas tarefas, alertar sobre tarefas atrasadas e estar habilitado a fazer um re-roteamento para outros usuários no caso de tarefas que estão muito atrasadas [WOR 2000].
- **Conectividade com Banco de Dados -** cada processo de *Workflow* ou utiliza informação dos bancos de dados para habilitar os usuários a tomadas de decisão, ou "alimenta" o banco de dados com nova informação [WOR 2000].
- **Documentos Anexados (*Document Attachments*) -** documentos são uma parte essencial dos processos de negócio. Uma solução de *Workflow* deve, todavia, fornecer um efetivo significado de documentos anexados para o *Workflow*, os quais são utilizados para suportar processos de negócio [WOR 2000].

3.5 Limitações dos Sistemas de Gerência de *Workflow*

De acordo com Georgakopoulos et. al. [GEO 95], os sistemas de gerência de *Workflow* em geral possuem algumas limitações:

- **Perda de interoperabilidade entre os Sistemas de Gerência de *Workflow*** - está diretamente ligada à carência ainda existente de padrões para os SGW. A falta de interoperabilidade causa dificuldades em relação à execução de serviços e funções dos SGW (API´s), protocolos e formatos de comunicação entre os SGW e entre SGW e outros aplicativos, e especificações de formatos de intercâmbio de modelos de *workflow* entre múltiplos SGW;
- **Perda de suporte para a interoperabilidade entre sistemas HAD (Heterogêneos, Autônomos e Distribuídos) ou entre SGW e sistemas HAD** - para *Workflows* que acessam sistemas de informação HAD, interoperabilidade completa entre os sistemas HAD, SGW e implementações de *Workflow* é importante pelas seguintes razões: (1) simplifica a implementação do *Workflow*, pois não é necessário código específico para aquele sistema, (2) permite a rápida implementação do *Workflow* em comparação àqueles que envolvem esta programação e (3) requer implementações mínimas para se ajustar às mudanças nas funcionalidades dos sistemas HAD, exigindo somente uma nova especificação de interface para o sistema HAD. Alguns SGW comerciais suportam limitada interoperabilidade entre algumas aplicações de escritório dependendo das necessidades de determinadas plataformas, interfaces ou sistemas operacionais;
- **Performance inadequada para alguns processos de negócio** - SGW comerciais geralmente suportam não mais do que umas poucas centenas de workflows ao dia. Alguns processos necessitam lidar com um grande número de *workflows*, implicando na não aplicação de SGW na gerência deste tipo de processo;
- **Perda de suporte para correção e confiabilidade** - na presença de concorrência e falhas. Para lidar com estes problemas é necessário que os SGW forneçam especificações de tarefas e ações de compensação (um conjunto de ações que atuam

como uma operação de desfazer - *undo*) e código para controle de concorrência e geração de *logs*. Exemplos de situações problemáticas em relação à correção e confiabilidade são: (a) diferentes *workflows* (em execução), que acessam recursos compartilhados, podem causar resultados inconsistentes se não forem modelados para acesso concorrente aos recursos, (b) interrupções abruptas do funcionamento de um processo de *workflow* exigem que o último estado consistente seja recuperado, através de operações de "desfazer" (*undo*), de ações de compensação e arquivos de *log*;

- **Fraco suporte de ferramentas para análises, testes e *debugs* de *Workflows*** (modelos de *workflow*) - existem poucos produtos que permitem a realização de análise, teste e depuração dos processos de *workflow*. Estas ferramentas são necessárias para simular execuções, estimar eficiência e determinar problemas da especificação e implementação dos *workflows*. Este tipo de ferramenta causa impacto direto sobre a rápida prototipação e fácil especificação e implementação dos modelos de *workflows*.

3.6 Alguns Sistemas de Gerência de *Workflow*

Nesta seção alguns dos SGW existentes no mercado atualmente serão brevemente descritos, dando-se uma ênfase maior ao produto "*Exchange 2000 Server*", visto que este é o SGW utilizado no protótipo de experimentação construído durante este trabalho.

3.6.1 *Ultimus Workflow*

O *Ultimus Workflow* é composto por quatro componentes, que são descritos a seguir [ULT 2000a].

Fluxograma de Organização (*Organization Chart*)

Para automatizar devidamente um *Workflow*, *Ultimus* tem uma capacidade de construção de fluxograma organizacional da empresa. Este fluxograma representa as funções (*job functions*) em um negócio e o relacionamento entre os participantes da empresa. Assim, uma determinada função pode ser associada a determinado empregado.

Ultimus Designer

O *Ultimus Designer* permite aos usuários criar ou modificar processos de negócio de *Workflow*. O *Ultimus Designer* realiza as seguintes funções:

1. Cria o mapa de processo.
2. Define eventos e condições para cada passo.
3. Define as planilhas que serão usadas no processo.
4. Cria o formulário a ser utilizado em cada passo.

5. Especifica os *Flobots* que serão usados no processo e que partes da informação serão utilizadas por cada *Flobot*.
6. Testa a aplicação de *Workflow* através de simulações.

Servidor do *Workflow Ultimus* (*Workflow Server*)

O servidor de *Workflow* do *Ultimus* é um pacote de software que controla a execução de um processo, assim como o seu progresso. A máquina (*Workflow server*) usa a definição de processo criada pelo *designer* para decidir quais ações deve tomar em cada passo. Uma vez que o processo é iniciado, a *engine* gera mensagens para os clientes envolvidos no processo. Estas mensagens instruem o que os clientes devem fazer, sendo que, assim que o cliente encerrar sua tarefa ele envia uma mensagem ao motor (*engine*) do *Workflow* informando sobre isto, o motor por sua vez toma as ações necessárias, assim como decidir qual deve ser o próximo passo do processo.

O Administrador *Ultimus*

Este administra o *Workflow* de negócios e fornece algumas características como:

1. Instalar e desinstalar um processo de *Workflow*.
2. Monitorar o estado do processo de *Workflow*.
3. Gerar estatísticas de performance.
4. Realizar funções administrativas.
5. Controlar o acesso de usuários.

3.6.2 Oracle *Workflow* Cartridge

O *Oracle Workflow Cartridge* [ORA 98] é um SGW integrado com aplicações *Oracle*, este permite que os dados transacionais sejam roteados em todo o processo de negócio, além do mais, os anexos das aplicações *Oracle* permitem capturar informações de todos os tipos de mídia - incluindo planilhas, imagens, áudio e vídeo - e associar estes com as informações de *Workflow* nas aplicações *Oracle*.

O *Oracle Workflow Cartridge* possui uma ferramenta gráfica chamada *Builder* que permite que novos processos de negócio sejam modelados e que os existentes sejam atualizados. Nesta ferramenta é possível modelar e automatizar processos de negócio. Pode se fazer a definição de laços (*loops*), desvios (*branchs*) para fluxos paralelos e então *rendezvous*, decompor para sub-fluxos, etc..

Este produto permite o uso completo da linguagem do servidor *Oracle8*, a PL/SQL, permitindo assim que automaticamente se processe informações mesmo para as mais sofisticadas regras de negócio.

No *Oracle Workflow Builder* é possível criar, visualizar ou modificar um processo de negócio com operações de arrastar e soltar (*drag*|*drop*). É possível se trabalhar com o modelo do *Workflow* em um nível resumido, podendo-se expandir as atividades dentro do *Workflow* quanto for necessário um nível maior de detalhes. A aparência do *Builder* da *Oracle* é mostrada na Figura 3.3.

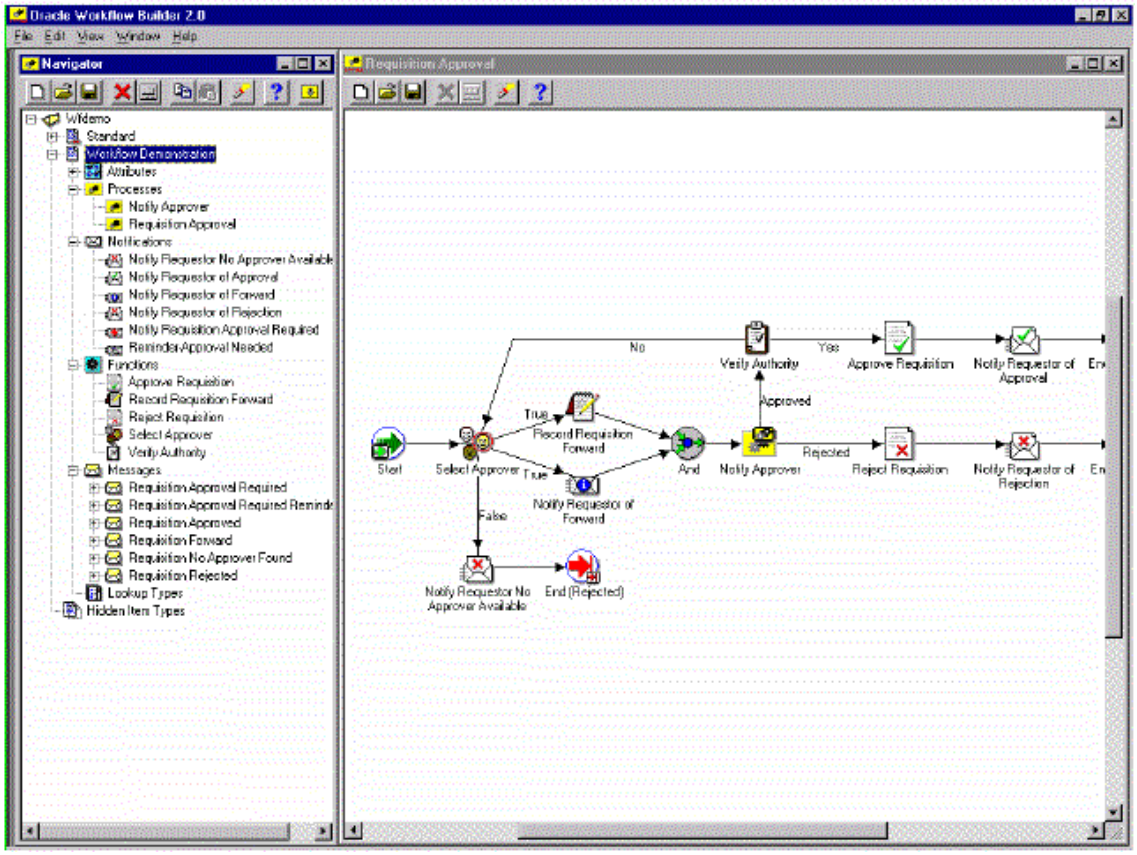

FIGURA 3.3 - Oracle *Workflow* Builder [ORA 98]

Uma máquina de *Workflow* integrada ao servidor *Orale8* monitora o estado do *Workflow* e coordena a distribuição das atividades durante o processo. Modificações no estado do *Workflow*, tais como o término de atividades, são sinalizadas para a máquina através de uma API da PL/SQL. Baseada sobre a flexibilidade das regras do *Workflow*, a máquina determina quais atividades são elegíveis para o processo, e então processa as mesmas.

O *Oracle* permite notificar usuários para que os mesmos realizem atividades que não podem ser automatizadas, tais como aprovação para requisições e ordens de venda. Notificações eletrônicas podem ser roteadas para usuários individuais ou para classes de usuários (usuários que desempenham os mesmos papéis - *roles*). Qualquer usuário, mesmo que não tenha fácil acesso às aplicações *Oracle*, podem ter acesso as notificações através de aplicações de e-mail. As notificações de e-mail contêm um documento anexo HTML onde o usuário pode obter as informações para as tarefas que deve realizar e responder a estas notificações.

Outra característica do *Oracle Workflow Cartridge* é que qualquer usuário que possua acesso a um *navegador Web* pode estar incluído no *Workflow*. Os usuários *Web* podem acessar uma página *Web* pré-configurada para ver suas notificações.

Este produto da *Oracle* também possui uma ferramenta de análise e monitoramento de performance, que pode ajudar o entendimento da performance do processo de *Workflow*. Com esta ferramenta é possível se identificar "gargalos" de performance e seguir a melhoria nos itens em movimento tais como ordens de compra, de venda e faturas através das transações dos sistemas [ORA 98].

3.6.3 MQSeries *Workflow* da IBM

O produto *IBM MQSeries Workflow* é baseado no modelo cliente-servidor, e os componentes do servidor podem estar instalados sobre uma variedade de plataformas incluindo OS/390, AIX, HP-UX, Sun Solaris, Windows NT e OS/2. O componente *Buildtime*, ou componente de definição, é suportado sobre a família *Windows* de sistemas operacionais, e é utilizado para definir os recursos de *Workflow* ligados ao processo, ao sistema como um todo, e ao pessoal necessários pelo ambiente do servidor, ou *Runtime*, para a execução sobre todas as plataformas *host* [DAV 2000].

O *MQSeries Workflow* fornece um sistema de automação de processo para a gerência de pessoas, dados, aplicações e processos de negócio em todo o âmbito de uma organização, incluindo parceiros externos via a *Internet* ou *Intranets* e *Extranets*. Com esta abordagem para *Workflow*, este produto fornece uma infra-estrutura de mensagens (*MQSeries Messaging*) e integração de aplicações (*MQSeries Integrator*) sobre a qual se constróem aplicações de *Workflow* a nível de produção.

MQSeries *Workflow* é SGW que permite a completa definição, gerência e execução de processos de negócio através da execução do software cuja ordem de execução é dirigida por uma representação de computador da lógica do *Workflow*. Neste contexto, um SGW tem três partes [MQS 2000]:

1. Uma função de definição de processo, através da qual o usuário modela o *Workflow*;
2. Uma função de controle que gerência o fluxo do trabalho em tempo de execução;
3. Interfaces que habilitam usuários humanos e aplicações realizar ações de negócio específicas dentro do *Workflow*.

Para a definição de um processo, o MQSeries utiliza um grafo direcionado para definição de processos. Isto ajuda a prevenir erros de modelagem como criação de laços sem fim, etc..

A Figura 3.4 mostra a visão de árvore com os processos que já foram definidos na parte a esquerda da janela da aplicação. Na direita, a visão de diagrama dos processos selecionados é mostrada; esta é a aparência do *Buildtime* do MQSeries da IBM.

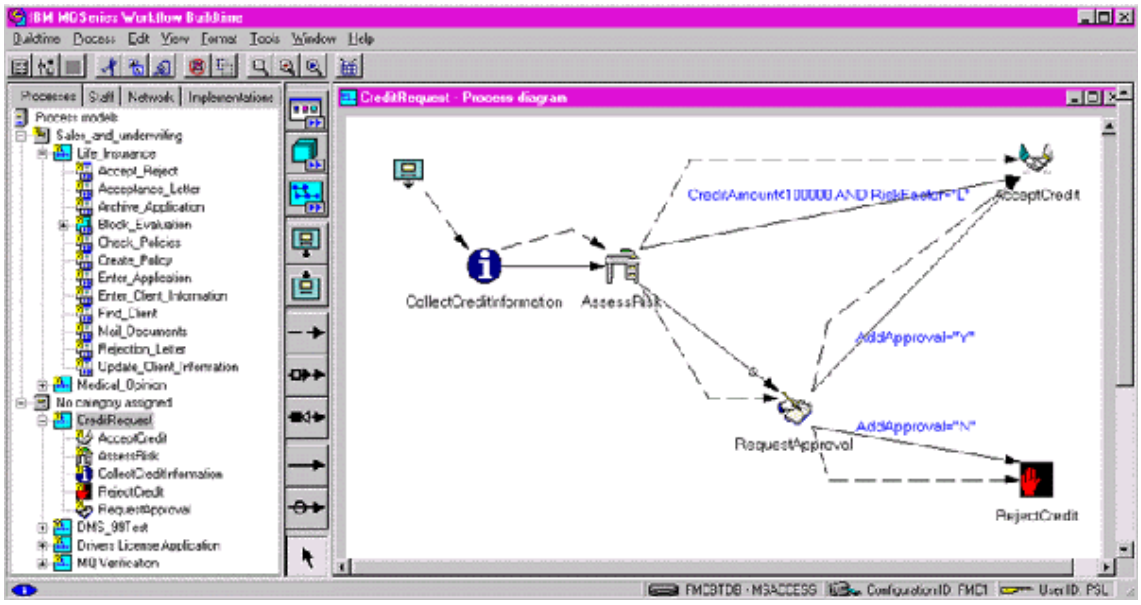

FIGURA 3.4 - Buildtime do MQSeries IBM [MQS 2000]

3.6.4 TIBCO - InConcert *Workflow*

TIB/InConcert modela, gerência e monitora em tempo real as tarefas realizadas como parte de um processo de negócio orientado a clientes, não interessando se estas tarefas são realizadas por usuários humanos ou agentes automatizados. InConcert fornece a habilidade de dinamicamente ajustar processos de negócios ativos e proativamente responder para as mudanças nas condições de negócio [TIB 2000].

O InConcert possui uma *engine* de processo que liga pessoas e aplicações de *Workflow*. Também possui uma interface com o usuário gráfica e intuitiva para o rápido projeto e prototipação de processos, esta interface pode ser visualizada na Figura 3.5. Este sistema suporta *Workflows* dinâmicos e estáticos através da *Internet*. Possui um monitoramento dos estados dos eventos em tempo real e ferramentas para gerência e medida de processos de negócio.

Esta ferramenta possui uma arquitetura aberta e orientada a objetos, que inclui um rico conjunto de interfaces entre programas com mais de 400 API's e regras de processos de negócio e é integrada completamente com outros produtos TIBCO.

A sua ferramenta gráfica *Process Designer* permite que os usuários projetem e modelem *Workflows* sem programação. O *Process Designer* inclui um número pré-definido de regras de processo de negócio que auxiliam na rápida prototipação dos processos de *Workflow*. As informações de projeto de processos são salvas em *Templates* e podem ser reutilizadas em qualquer parte da organização.

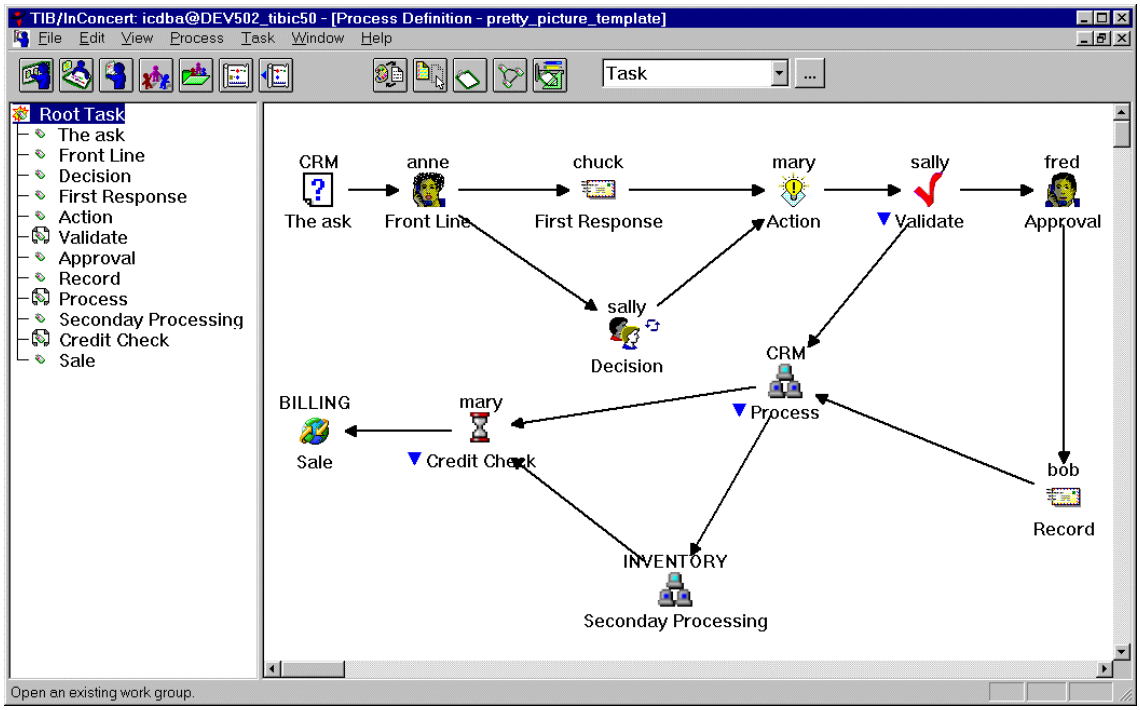

FIGURA 3.5 - Interface com o usuário do InConcert [TIB 2000]

InConcert oferece uma extensa biblioteca de Clientes API que podem ser usadas na integração de tarefas de *Workflow* para dentro de aplicações existentes e permitem a fácil criação de aplicações customizadas. Além disso, no intuito de suportar as interfaces padrões da indústria (C, C++, OLE, Java), InConcert inclui um suporte à diretório LDAP para ajudar a diminuir a carga de administração de usuário através de um completo *framework* de *Workflow*.

3.6.5 BizFlow da HandySoft

BizFlow-2000 da *HandySoft* fornece uma solução de *Workflow* a nível de empresa para definir, modelar e automatizar processos de gerência e de negócios. Esta ferramenta inclui um *form designer* que converte qualquer formulário (*form*) ou documento para um formato eletrônico e pode ligar *forms* para um ou mais bancos de dados relacionais por meio de ODBC ou ADO para “ler” dados ou “escrever” dados de/para o banco de dados.

Esta ferramenta também possui um *Workflow designer* que utiliza uma interface gráfica para definir e modelar negócios e processos de gerência e de negócios, então incorporar regras de negócio para automatizar o *Workflow* resultante [BIZ 2000].

BizFlow é composto por um conjunto de componentes integrados que trabalham juntos para fornecer uma solução de *Workflow* baseada na Web para qualquer aplicação. Os seus componentes são [BIZ 2000]:

- **BizFlow-2000 Server** – este contém a *engine* do *Workflow*. Esta fornece a lógica de software para gerir e controlar o *Workflow*. Pode rodar em UNIX, Windows NT ou Linux e suporta múltiplos servidores. O servidor pode realizar a interface com

qualquer banco de dados relacional. BizFlow-2000 suporta os padrões de *Workflow* definido pela WFMC [WMC 2001].

- **BizFlow-2000 Web Client** – este habilita o participante do *Workflow* a iniciar e executar os processos de *Workflow* a partir de qualquer localização utilizando um *navegador* Web. O usuário organiza as funções do *Workflow* por *folders* para facilitar o gerenciamento dos processos. O usuário pode selecionar notificações para cada passo do *Workflow* que fornecem uma mensagem como cada passo é realizado. Tarefas com maiores prioridades são codificadas com outras cores e listadas em primeiro plano nas listas de trabalhos (*worklists*). Usuários também podem atachar comentários eletrônicos para o *Workflow* para que possam ser vistos por outros usuários.

- **BizFlow-2000 *Workflow* Administrator** – este opera em um ambiente cliente-servidor puro com o software cliente que se conecta diretamente com o servidor do BizFlow. Este componente fornece as capacidades poderosas para os administradores do sistema e gerentes para gerir todos os aspectos do SGW. O *Administrator* fornece o seguinte conjunto de ferramentas:

  - ***Designer* gráfico de processos (*Graphical Process Designer*).** Este tem a capacidade de projetar novas definições de processo e modificar definições de processos existentes. Esta ferramenta fornece uma interface com usuário gráfica que possuí características de *drag* e *drop* que não necessitam habilidades de programação para definir, modelar e automatizar processos de *Workflow*. A Figura 3.6 descreve esta interface.

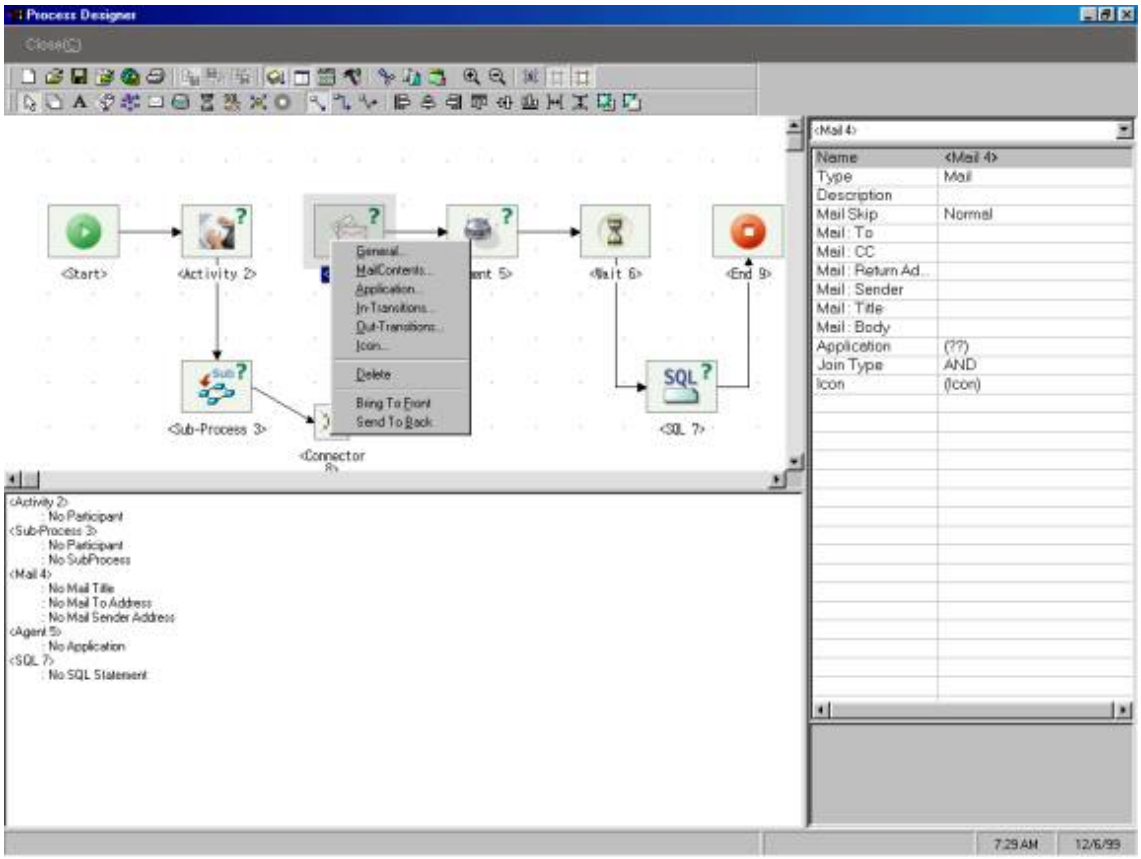

FIGURA 3.6 - Process Designer do BizFlow [BIZ 2000]

- ***Graphical BizForm Designer***: Projeta novos formulários eletrônicos e documentos. Modifica formulários já existentes e documentos de ligação para múltiplos bancos de dados relacionais. BizForm Designer é um programa com ricas características que fornece capacidades de projetar qualquer tipo de formulário ou documento e então ligar as células sobre o formulário para um ou mais bancos de dados usando ou ODBC ou ADO. BizForm possui um *Designer* usado para a construção de formulários e um *Navegador* usado pelos usuários para visualização dos formulários como parte do *Workflow*. Esta ferramenta suporta OLE, planilhas do Excel, páginas Web ativas ou muitas outras aplicações de terceiros que podem ser integradas dentro do formulário. Armazena dados em formato XML separado dos formulários, reduzindo a quantidade de informação que deve ser carregada para o cliente Web quando executando o *Workflow*.
- ***Organization Chart Manager*** - Define a organização, adicionando e apagando departamentos e usuários. Estabelece arquivos de assinaturas eletrônicas. Gerência os *passwords* dos usuários, assim como os papéis e privilégios dos mesmos. Estabelece grupos de usuários e configura endereços de e-mail dos usuários.
- ***Application Manager*** - Define as aplicações a serem utilizadas em conjunto com o *Workflow*. Este suporta vários tipos de aplicações como: BizForm, JAVA, Script, aplicações cliente, aplicações servidor, planilhas Excel, documentos Word, páginas Web e URLs.
- ***Process Analyzer*** - BizFlow-2000 inclui o *Process Analyzer*, o qual produz relatórios customizados em formatos de fluxogramas ou de tabelas. Esta característica possibilita que aos gerentes medir a performance do *Workflow*.

- **BizFlow-2000 Developer** - Uma das características mais marcantes do BizFlow é o *Developer*. Este possui APIs e controles que podem ser utilizados com qualquer ferramenta de 4ª geração (4GL) tais como MS Visual Basic e Borland Delphi para o desenvolvimento de aplicações customizadas. Com o *developer*, o BizFlow serve como uma plataforma de desenvolvimento de aplicações que oferece flexibilidade para que os usuários possam ter recursos para automatizar processos de *Workflow* complexos.

3.6.6 Microsoft Exchange 2000 Server

Exchange 2000 Server (E2K) permite a construção de aplicações que modelam e automatizam processos de negócio [MAR 2000]. As principais características do E2K são o *Active Directory* (AD), *Web Storage System* (WSS), capacidade de endereçamento por URL, várias tecnologias de acesso a dados como ADO, CDO XML, suporte a eventos e lógica de *Workflow* [MAR 2000] [RIZ 2000].

O AD, que faz parte do sistema operacional Windows 2000, armazena informação sobre recursos de rede e torna estas informações disponíveis para os usuários da rede e as aplicações. WSS permite o armazenamento de qualquer tipo de

dados. Endereçamento por URL significa que o acesso aos dados pode ser feito via *Internet*, possibilitando o armazenamento distribuído de informação [MAR 2000].

Eventos síncronos e assíncronos reagem da mesma maneira para as requisições de dados. Uma requisição de dados pode ser um novo recurso, uma requisição de modificação, exclusão, cópia ou movimento de recurso. Eventos síncronos disparam antes que uma requisição de dados seja confirmada (*committed*) para o WSS, enquanto que eventos assíncronos disparam depois que uma requisição é confirmada. Eventos do sistema não reagem às requisições de dados, mas, ao invés disso, as mesmas disparam quando um BD inícia ou para em um determindado horário [MAR 2000].

O *Exchange 2000 Server* foi projetado especificamente para o sistema operacional Windows 2000 Server, fornecendo confiabilidade avançada, escalabilidade e performance com baixo custo, os quais são derivados através de um gerenciamento unificado de serviços de mensagem, colaboração e recursos de rede [MIC 2000].

O armazenamento de dados no E2K é feito através do WSS. Cada WSS é organizado como um sistema de arquivos tradicional, isto é, uma hierarquia de pastas. As pastas podem armazenar qualquer ítem, desde ítens padrão do E2K como contatos, mensagens, agendamento de reuniões (*appointments*) até arquivos mais complexos como páginas ASP (páginas de servidor ativas - *Active Server Pages*), documentos Microsoft Word e Excel e modelos de ferramentas como Rational Rose (arquivos ".mdl") [RAT 2001], etc.. Pastas dentro de pastas também são possíveis.

Exchange 2000 suporta um grande campo de atividades colaborativas, incluindo capacidades de agendamento de grupos, grupos de discussões e pastas de grupos. Com a indexação e procura de conteúdo embutido (*built-in*) (as consultas podem ser adaptadas de acordo com a aplicação), usuários podem encontrar e compartilhar informação rapidamente. Juntamente com o WSS, E2K combina poderosas ferramentas de *Workflow* com padrões *Web* como XML e HTTP, resultando em plataformas para aplicações *Web* de alta performance tais como gerência de conhecimento e comércio eletrônico (*e-comerce*) [MIC 2000].

- **Fácil acesso à informação:** informação armazenada no WSS pode ser acessada através de uma variedade de softwares cliente (*Outlook*, clientes de e-mail, *Windows Explorer*, qualquer *navegador* e muitas aplicações *Windows*).
- **Integração com o Office 2000:** o WSS provê um modelo consistente e um conjunto de ferramentas para gerência conjunta de *e-mail* e documentos.
- **Construção de Aplicações Web com ferramentas e habilidades existentes:** o WSS inclui serviços *built-in* para construção de aplicações de alta performance, incluindo suporte para XML, *World Wide Web Distributed Authoring and Versioning* (WebDAV) e *Active Server Pages* (ASP), formulários de acesso a dados customizados, eventos de lógica de negócios e componentes reutilizáveis. Tecnologias de acesso a dados, como OLE DB e Microsoft® *ActiveX Data Objects* (ADO), proporcionam que os desenvolvedores possam aproveitar suas funcionalidades e aplicações existentes quando construindo aplicações WSS. Aplicações WSS podem ser construídas com ferramentas de desenvolvimento tais como *MS Visual Studio*, *MS FrontPage 2000*, e *MS Office Developer Edition* [MIC 2001].

- **Automatização de Processos de Negócio:** uma *engine* de *Workflow* embutida (*built-in*) fornece uma alta performance, confiabilidade e plataforma segura para aplicações de *Workflow* e de "*tracking*" a nível departamental e de empresa. Os sistemas de *Workflow* são projetados utilizando-se uma ferramenta visual, o *Workflow Designer*.
- **Aumenta a efetividade e competitividade de soluções baseadas na Web:** através da forte integração do E2K e *Internet Information Services*, desenvolvedores podem colocar ambas suas aplicações e conteúdos de aplicações em um único local em E2K. O suporte nativo para XML e HTTP fornece meios para que o E2K preencha facilmente os requisitos para a infraestrutura necessária para *e-comerce* [MIC 2000a].

Exchange 2000 provê extensas oportunidades para a construção e emprego de soluções de negócio adaptadas (*custom*) tais como aplicações *Workflow* e aplicações nativas baseadas na Web. Estas aplicações permitem conduzir os processos de negócio, sem a preocupação de o quão grande ou distribuído são os negócios. Características como o WSS, biblioteca de programação com objetos CDO (*Collaboration Data Objects*), um motor (*engine*) de *Workflow* com ferramenta de projeto visual, um poderoso modelo de eventos e integração com ferramentas fornecem meios para a rápida construção de soluções para os processos de negócio [MIC 2000a].

- E2K fornece uma máquina de *Workflow* de construção customizada (*built-in* - possibilita a construção de processos de *Workflow* específicos para as aplicações) que permite a construção de soluções de *Workflow* robustas.
- *Workflow* Designer for Exchange 2000 provê uma ferramenta de modelagem gráfica para os desenvolvedores modelar e implementar suas soluções de negócio. Com o uso desta ferramenta é possível se criar rápida e facilmente soluções de *Workflow*.
- Melhoramentos no modelo de eventos do E2K permite que os projetistas criem aplicações utilizando eventos síncronos que usam lógica de negócios para automatização dos processos de negócio.
- E2K suporta completamente OLE DB 2.5, possibilitando uma nova geração de aplicações de negócio que integram E2K, Microsoft® SQL Server e outros produtos padronizados com o OLE DB. Estas interfaces possibilitam que desenvolvedores utilizem facilmente o E2K em suas aplicações.

E2K foi o servidor de colaboração escolhido para o experimento parcialmente por que algumas empresas já possuem este como um servidor de correio eletrônico. Entretanto, o mesmo também poderia ser utilizado como um SGW, não necessitando, assim, modificar a principal interface em uso se fosse adquirida uma nova ferramenta. De fato, qualquer Sistema de Gerência de *Workflow* (SGW) poderia ser utilizado, desde que este forneça generalidade no processo de modelagem, no sentido que possa ser utilizado em muitos domínios, incluíndo desenvolvimentos técnicos.

3.6.6.1 *Workflow* Designer for Exchange 2000 Server

A ferramenta para definição visual de processos de *Workflow* do E2K é a *Workflow Designer*. *Workflow Designer* provê uma interface gráfica para a modelagem da lógica do *Workflow*. As ações e condições do processso de *Workflow* são implementadas com a linguagem de script *VBScript*. As tecnologias ADO e CDO [MIC 2001a] são os principais dispositivos de acesso e armazenamento de dados e são utilizados dentro de códigos em *VBScript* [VBS 2001]. Um processo de *Workflow* é associado com um pasta pública (que pode ser acessada pelos usuários da aplicação), onde o E2K monitora qualquer evento que ocorra dentro desta pasta e, de acordo com as condições definidas na ferramenta *Workflow designer*, dispara as ações associadas [MAR 2000] [MWY 2000] [RIZ 2000].

A interface da ferramenta *Workflow* Designer pode ser visualizada na Figura 3.7. As ações do processo de *Workflow* são definidas no campo "*Action Script Procedure*". Neste campo, chamadas a subrotinas e funções podem ser feitas, estas subrotinas ou funções podem ser definidas através do editor existente no campo "*Shared Script*" ou através de um arquivo texto o qual deve ser referenciado. A condição para que um código de ação (*Action Script Procedure*) seja executado é definida no campo "*Condition Expression*".

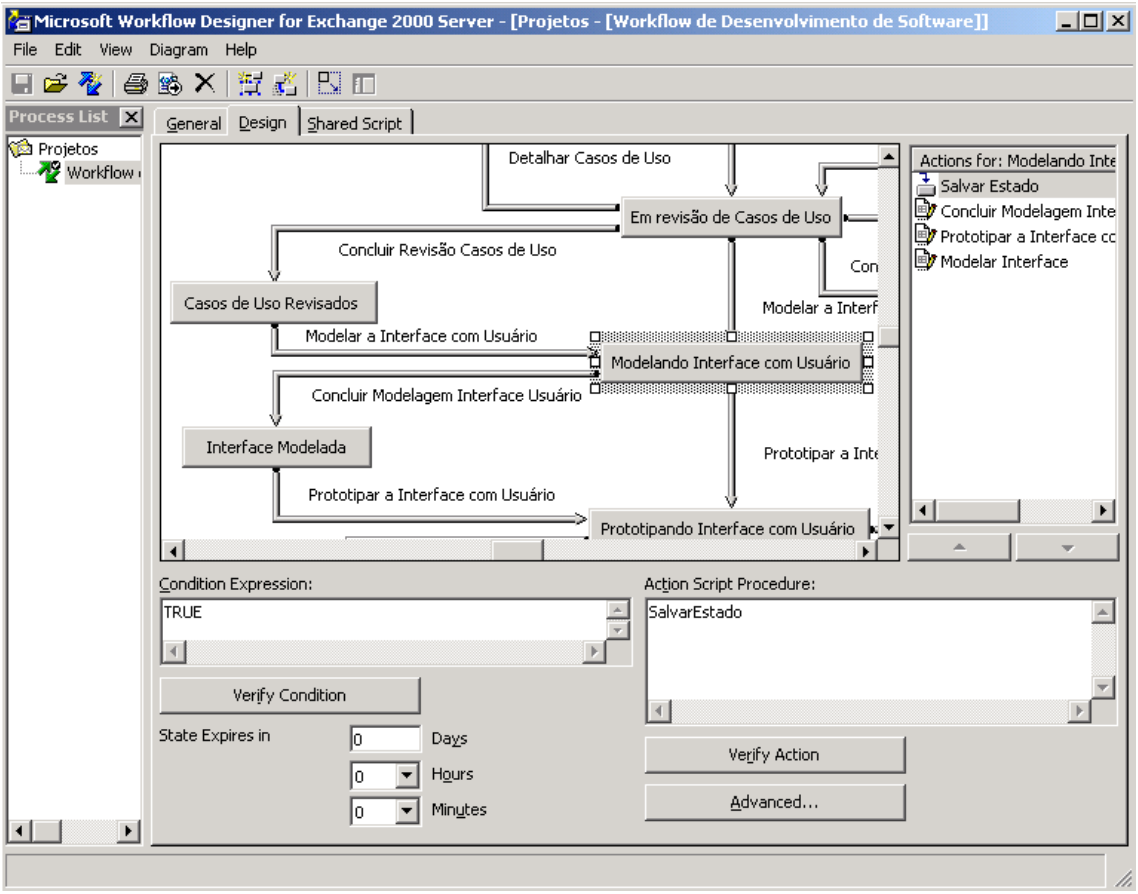

FIGURA 3.7 - Interface do Microsoft *Workflow* Designer for Exchange 2000 Server

3.6.6.2 Web Storage System

O *Web Storage System* (WSS) é uma tecnologia de banco de dados que pode ser utilizada para armazenar, compartilhar e gerenciar dados heterogêneos, como mensagens de e-mail, páginas Web, arquivos multimídia e documentos Microsoft® Office 2000. O WSS é organizado, parecido com sistemas de arquivos tradicionais, em uma hierarquia de pastas. Cada pasta no WSS pode conter qualquer número de itens, incluindo outras pastas. O acesso aos itens pode ser feito através de diversos protocolos e interfaces de programação de aplicações (APIs – *Application Programming Interfaces*), incluindo HTTP e World Wide Web Distributed Authoring and Versioning (WebDAV). Microsoft® ActiveX Data Objects 2.5, OLE DB 2.5, CDO para Microsoft® Exchange 2000 Server (CDOEX), MAPI (através de sistema de arquivos) e utilizando outros protocolos padrões das industrias [LAA 2001].

O WSS armazena qualquer tipo de dados e também permite o acesso para estes dados de diferentes clientes que utilizam diferentes protocolos. Como é possível armazenar qualquer tipo de dados no WSS, um mecanismo de extensível deve existir para descrever e complementar os dados. Os itens de dados no WSS pode ser adaptados de acordo com as aplicações através da adição de propriedades customizadas a estes itens [LAA 2001].

Estas propriedades customizadas podem ser utilizadas por desenvolvedores dentro de suas aplicações *Workflow*. Estes desenvolvedores podem escrever códigos adaptados (*sink*) que o Exchange 2000 executa quando um evento em particular é disparado em uma pasta específica. Estes códigos podem ser escritos em Microsoft® *VBScript* [VBS 2001], Visual Basic ou C++. O processo de armazenamento (*store.exe*) passa parâmetros para o *sink* que permite determinar porque um evento foi disparado. Eventos síncronos ocorrem no caminho de transações, assim, os mesmos podem fazer com que o Exchange pare uma transação [LAA 2001].

Para desenvolver aplicações de *Workflow*, é necessário se compreender como o motor de *Workflow* (*Workflow engine*) utiliza a arquitetura do E2K e como utilizar objetos de CDO (*Collaboration Data Objects*) e a ferramenta *Workflow Designer* para habilitar processos sobre itens dentro do WSS [LAA 2001].

3.6.6.3 Compatibilidade entre o Exchange Server e a Arquitetura de SGW proposta pela WFMC

Esta seção avalia o E2K comparando com a estrutura genérica de SGW proposta pela *Workflow Management Coalition* - *WFMC* [WMC01] - de acordo com os seus principais componentes. Esta comparação visa avaliar o nível de padronização do E2K em relação a WFMC.

Ferramenta de Definição (*Definition Tool*) - a ferramenta *Workflow Designer* permite a definição de processos de negócio, através de uma interface gráfica (que possui uma notação semelhante aos diagramas de estados da UML [FOW 2000] [LAR 2000] [BOO 2000]) que permite a inserção de estados e ações associadas a estes estados. Estas ações podem ser disparadas segundo condições pré-definidas e em momentos específicos, como: na saída de um estado durante uma transição, quando na entrada em um novo estado, quando um item é salvo na pasta pública, etc. A *Workflow Designer* é limitada em alguns aspectos no sentido de definição de processos, pois muitas funcionalidades como o envio de mensagens, recuperação/modificação/inserção

de dados nas pastas públicas, criação de novos itens de atividades nas listas de tarefas dos participantes do *workflow*, etc., devem ser implementadas utilizando-se procedimentos escritos na linguagem de *script VBScript*. As transições de estados são automaticamente feitas pela ferramenta, necessitando somente modelar os estados e indicar as transições entre os estados junto com suas condições de transição (troca de estados).

Definição de Processo (*Process Definition*) - a ferramenta *Workflow Designer* é utilizada na construção dos processos de *Workflow* dentro de uma pasta pública do Servidor *Exchange*. A *Workflow Designer* cria os arquivos necessários contendo a descrição do processo de *workflow* e armazena estes em uma pasta pública; o Servidor *Exchange*, de posse destes arquivos, controla os eventos que ocorrem dentro desta pasta e, de acordo com as condições e restrições definidas no modelo de *workflow*, dispara ações associadas e faz as devidas transições de estados.

Serviço de Execução de *Workflow* (*Workflow Enactment Service*) - o motor de *Workflow* (*Workflow engine*) controla as modificações dos itens, os quais são armazenados em uma pasta, reagindo contra os eventos e disparando as ações apropriadas. Os dados relevantes e acessados do *Workflow* são armazenados no WSS.

Lista de Tarefas/Trabalho (*Work List*) - o assinalamento de itens de trabalho para os participantes do processo deve ser feito através da implementação de ações na linguagem *VBScript*. Estas ações podem ser utilizadas para enviar e-mail ou produzir listas de tarefas associadas aos participantes.

Gerente de Listas de Trabalho e Interface Usuário (*Worklist Handler* e *User Interface*) - os dados do E2K podem ser visualizados pelo uso do *Microsoft Outlook 2000*, *Microsoft Outlook Web Access* e por formulários do WSS [MAR 2000]. É necessária a criação de interfaces de acesso e manter as listas de tarefas pela implementação de documentos ASP.

Aplicativos (*Applications*) - a ferramenta *Workflow Designer* não suporta normalmente a ativação de ferramentas externas. No prototipo experimental descrito no Capítulo 6, a ativação de ferramentas é realizada por meio de links em documentos HTML, que executam *Scripts* em *VBScript* e *JavaScript* [VBS 2001] que ativam ferramentas externas. O *Microsoft Outlook 2000* é completamente integrado com as ferramentas do *Microsoft Office* [MIC 2001], permitindo o uso de documentos Word e planilhas Excel como disposivos base para a colaboração. A ativação de outros aplicativos (como as contidas no *suite Rational* [RAT 2001]), como editores de diagramas, compiladores, ferramentas de teste e tantas outras utilizadas no processo de desenvolvimento de software, necessitam, como já mencionado, de funções implementadas em linguagens de *Script* que ativem estas ferramentas externas.

Supervisão - Administração e Monitoramento (*Administration & Monitoring*) - Para criar e registrar aplicações de *Workflow* no E2K o usuário deve estar registrado no servidor. A definição de processos é feita em uma ferramenta gráfica, permitindo a especificação de estados de *Workflow*, transições de estados e suas ações e condições associadas. O E2K não possui nenhum dispositivo automático para monitoramento dos processos de *workflow*, por outro lado, muitas tarefas podem ser programadas - como a inserção e recuperação de dados do WSS, envio de mensagens, ativação de ferramentas, etc. - permitindo a construção de funções que capturem

informações que possam fornecer aos gerentes de *workflow* meios de monitoramento dos processos.

3.6.6.4 Limitações do *Microsoft Exchange 2000 Server*

Ao longo do trabalho foi possível notar algumas limitações inerentes ao SGW *Exchange 2000 Server* (E2K). Estas limitações serão descritas a seguir:

- A nesessidade de se programar a criação e envio de mensagens de e-mail – todas as mensagens enviadas pelo motor de *workflow* do E2K foram criadas através de códigos em *VBScript* [VBS 2001] e utilização da biblioteca CDO [MAR 2000] [MIC 2001a] [RIZ 2000]. A Figura 3.8 demonstra um exemplo de função para criação e envio de mensagens de e-mail;

```
Sub SendMessageHTML (strTo, strSubject, strBody)
    Set oMsg = CreateObject("CDO.Message")
    oMsg.To = strTo
    oMsg.From = WorkflowSession.Sender
    oMsg.Subject = strSubject
    oMsg.HTMLBody = strBody
    oMsg.Send
    Set oMsg = Nothing
End Sub
```

FIGURA 3.8 - Função VBScript para Criação de Mensagens de E-mail

- A disponibilização de listas pessoais de atividades (*to-do lists*) através de programação VBScript – as atividades (tarefas) associadas a cada usuário do *workflow* devem ser criadas dinamicamente através da consulta dos itens de "Atividades" (descritos no Capítulo 6) criados dentro do E2K. Esta consulta é realizada através de expressões SQL [MAR 2000] [RIZ 2000] [SIL 99] que são inseridas em código ASP [MAR 2000] [RIZ 2000]; estas consultas são processadas e os seus resultados são disponibilizados através de HTML. A Figura 3.9 mostra um exemplo de consulta SQL, inserida em código *VBScript* (documentos ASP), para a consulta sobre as atividades associadas a um determinado usuário. A consulta é feita de acordo com o endereço de e-mail do usuário – *Session("usuario")* - e com o *Content-Class* dos itens da pasta – *'gpmgt:content-classes:atividade'*;

```
strSQL = "Select " & _
  AddQuotes("DAV:displayname") & ", " & _
  AddQuotes("DAV:contentclass") & ", " & _
  AddQuotes("gpmgt:DescAtividade") & ", " & _
  AddQuotes("gpmgt:DescProjeto") & ", " & _
  AddQuotes("gpmgt:Template") & ", " & _
  AddQuotes("gpmgt:Guideline") & ", " & _
  AddQuotes("gpmgt:Ferramenta") & ", " & _
  AddQuotes("gpmgt:IdAtividade") & ", " & _
  AddQuotes("DAV:href")
strSQL = strSQL & _
  " FROM SCOPE('SHALLOW traversal of " & _
  AddQuotes(urlQueryFld) & "')"
‘ urlQueryFolder é a URL da pasta pública a ser consultada
‘AddQuotes() é uma função que duplica as aspas de uma string
strSQL = strSQL & _
  " WHERE " & AddQuotes("gpmgt:Resp") & _
  " = '" & Session("usuario") & "' AND " & _
  AddQuotes("DAV:contentclass") & _
  " = 'gpmgt:content-classes:atividade' "
strSQL = strSQL & _
  " ORDER BY " & AddQuotes("gpmgt:DescProjeto")
```

FIGURA 3.9 - Exemplo de Consulta SQL sobre os Itens de "Atividades"

- Impossibilidade de se modelar, em nível conceitual, a concorrência entre atividades – não é possível, ao menos em nível conceitual, se modelar atividades concorrentes, visto que o *Microsoft Workflow Designer* utiliza uma notação parecida com diagrama de estados da UML [FOW 2000] [LAR 2000] [BOO 2000] e esta não permite que um item em determinado estado, após a avaliação de uma condição para troca de estado, passe deste estado para múltiplos estados (uma operação tipo *fork*). Este tipo de ação tem que ser simulado pela criação de múltiplos itens de atividades (isto é, a criação de várias atividades em paralelo) em uma mesma transição de estados;
- A ativação de ferramentas externas não é realizada automaticamente pelo E2K – para a ativação de ferramentas externas é necessário que funções escritas em *JavaScript* ou *VBScript* [VBS 2001] sejam implementadas. Ferramentas do pacote *Office* (como o *Word* e *Excel*) da *Microsoft* [MIC 2001] podem ser ativadas naturalmente, mas ferramentas de outros fabricantes devem ser ativadas pela programação de *scripts*;
- A falta de portabilidade do *Exchange 2000 Server* (E2K). O E2K faz uso de diversos serviços do *Windows 2000 Server*, e isto possibilita a utilização de muitos serviços oferecidos por este sistema operacional. Entretando, esta dependência de sistema operacional causa problemas de portabilidade e acesso. As aplicações definidas no E2K só podem ser acessadas, pelo menos da maneira esperada, a partir de máquinas cliente que possuam os sistemas operacionais *Microsoft Windows 2000* ou *Windows NT 4.0*, e utilizem como navegador *web* o *Microsoft Internet Explorer*. Máquinas que utilizam outro sistema operacional (como *Windows 98/95* e *Linux*) ou outro navegador *web* (como *Netscape*) não conseguem acessar as páginas das aplicações do E2K. Este problema pode restringir a utilização do E2K como SGW dentro das organizações.

3.7 Comparação entre os SGW Apresentados

A Tabela 3.1 apresenta um quadro comparativo entre os SGW apresentados nesta seção. Esta comparação ressalta algumas caracteríticas importantes, segundo a WFMC [WMC 2001] e alguns dos próprios fabricantes destes produtos de *workflow*, dos Sistema de Gerência de *Workflow*.

A escolha do *Exchange 2000 Server* (E2K), como o Sistema de Gerência de *Workflow* (SGW) a ser utilizado no protótipo de experimentação, também se deve a outro fator (além do mesmo já ser utilizado como servidor de correio eletrônico por algumas organizações), que é o fato do E2K ter sido o único SGW disponível para utilização do Grupo Amadeus [AMA 2002] durante a realização deste trabalho.

TABELA 3.1 - Comparação entre os SGW apresentados (Continua)

Característica a ser considerada	Características Individuais dos SGW
Ferramenta de Definição de Processo	**Ultimus** - possui uma ferramenta gráfica para modelagem dos processos, o *Ultimus Designer*. A notação desta ferramenta se assemelha a um diagrama de atividades da UML. **Oracle Cartridge** – possui a ferramenta gráfica *Builder* para modelagem de processos, que propicia a descrição das atividades a partir de um diagrama de atividades. Este diagrama utiliza ícones que representam elementos do diagrama como: atividades, condicionais, símbolos iniciais e finais, etc. **MQSeries IBM** – a ferramenta *Buildtime* permite a modelagem gráfica das atividades do *workflow*, utilizando os ícones de sua barra de ferramentas de modelagem. **InConcert TIBCO** – possui uma ferramenta gráfica chamada *Process Designer* e utiliza uma notação parecida com diagrama de atividades da UML. Possui um conjunto de regras de processo de negócio pré-definidas para rápida prototipaçõa de processos de *workflow*. **BizFlow HandySoft** – a ferramenta *Process Designer* permite a modelagem de *workflows*. **Microsoft Exchange 2000 Server** - possui a ferramenta gráfica *Microsoft Workflow Designer for Exchange 2000 Server*. A notaçõa desta ferramenta é assemelhada aos diagramas de estados da UML. Possui campos de programação de condições e ações através de *scripts* em *VBScript*. O modelo de processo tem o seu controle sobre os eventos de uma determinada parta definida dentro do *Exchange 2000 Server*.
Arquitetura da Aplicação	**Ultimus** – utiliza uma arquitetura em três camadas (cliente/servidor/banco de dados) e possui 4 componentes básicos : *Designer* (modelagem dos processos), *Administrator* (faz a gerência e monitoramento de execução, monitorando processo, gerando estatísticas, controlando acesso, etc.), *Org Chart* (define os papéis, e a hierarquia entre estes papéis, desempenhadas pelos usuários do SGW) e *Workflow Server* (se encarrega da execução a andamento do processo de *workflow*). **Oracle Cartridge** – integrado com as aplicações Oracle, utilizando o Banco de Dados *Oracle*. As notificações de novas atividades são feitas através de API's da linguagem PL/SQL da Oracle. Permite a modelagem de papéis através do *Suite Oracle*. **MQSeries IBM** – arquitetura em três camadas. Infra-estrutura composta pelo *MQSeries Messaging* (fornece os serviços de mensagens), *MQSeries Integrator* (integra aplicações externas ao processo de *workflow*) e *MQSeries Workflow* (fornece o ambiente de automação do processo e fornece dispositivos de gerência e monitoramento). **InConcert TIBCO** – possui uma arquitetura de três camadas. É uma arquitetura aberta e orientada a objetos que possui mais de 400 API's paraintegração com outras aplicações. **BizFlow HandySoft** – possui os seguintes componentes: *Server* (contém o motor de *workflow*, suporta os padrões da WFMC), *Web Client* (fornece a interface com os usuários do *workflow*, possibilita a visualização de listas de tarefas, inicialização e dinalização de tarefas, etc.), *Administrator* (é o principal componente do BizFlow, fornece os dispositivos de gerência e monitoramento, permite a modelagem de processos de negócio, possibilita a definição da hierarquia da organização, aplicações externas a serem integradas, formulários de acesso a informações do Banco de Dados, etc.) e *Developer* (permite a programação de procedimentos especiais para serem executados pelo motor de *workflow*, esta funcionalidade pode ser utilizada a partir de alguma ferramenta de programação de 4ª geração como *MS Visual Basic* ou *Borland Delphi*). **Microsoft Exchange 2000 Server** - utiliza o WSS como banco de dados e está totalmente integrado ao *Windows 2000 Server*. Utiliza as bibliotecas ADO, CDO e OLE DB para acesso, criação e atualização dos seus itens (itens do Exchange). Os itens do *Exchange* permitem a criação de atividades, usuários, equipes de revisão, etc. pela definição de novas propriedades associadas aos itens criados a partir da biblioteca CDO. Suporta XML, WebDAV e ASP.
Monitoramento de Execução do Workflow	**Ultimus** - o monitoramento de execução do *workflow* é realizado pelo *Administrator*. **Oracle Cartridge** – engine do workflow, integrada ao servidor *Oracle8* monitora o estado do *workflow*. Possui ferramenta especializada para monitoramento e análise de performance do *workflow*. **MQSeries IBM** - *MQSeries Workflow* realiza a definição, gerência e execução do processo de *workflow* **InConcert TIBCO** - Possui monitoramento do estado do *workflow* e ferramentas para gerência e medida de processos de negócio. **BizFlow HandySoft** – possui o *Process Analyzer*. Ferramenta para avaliação de performance e geração de relatórios. **Microsoft Exchange 2000 Server** - não possui uma ferramenta de monitoramento dos processos. É necessário que os dispositivos de monitoramento sejam implementados através de programação de *scripts* que armazenem informações relativas a monitoramento.

TABELA 3.1 - Comparação entre os SGW apresentados (Continuação)

Modificação dinâmica da definição de processo de workflow	**InConcert TIBCO** - habilidade de dinamicamente ajustar processos de negócios ativos e proativamente responder para as mudanças nas condições de negócio. **BizFlow HandySoft** – idem 1º. **Microsoft Exchange 2000 Server** – permite a modificação dinâmica do processo de *workflow*, após a modificação os processos ativos passam a obedecer ao novo modelo de processo de *workflow*.
Integração com outras ferramentas na execução do processo de workflow	**Ultimus** - permite a integração com outras aplicações através dos *Flobots* - que são denominados agentes de automação. **Oracle Cartridge** – permite o uso da linguagem PL/SQL. Possui integração com outras aplicações da *Oracle*. **MQSeries IBM** - a integração com outras aplicações é realizada através do *MQSeries Integrator*. **InConcert TIBCO** – biblioteca de APIs para integração de tarefas de *workflow* para dentro de aplicações. **BizFlow HandySoft** – possui o *Aplication Manager* para realizar a integração com outras ferramentas. **Microsoft Exchange 2000 Server** - a integração com ferramentas de terceiros necessita ser programada através de *scripts*. A integração com produtos *Microsoft* é permitida normalmente pelo E2K.
Programação de Ações Especiais e/ou Adaptadas às Aplicações	**InConcert TIBCO** - por ser totalmente programável, permite a criação de aplicações adaptadas a aplicações específicas. **BizFlow HandySoft** – possui o *Developer* que permite que aplicações adaptadas sejam construídas com base em qualquer linguagem de 4ª geração (4GL) como *Visual Basic* ou *Delphi*. **Microsoft Exchange 2000 Server** - o E2K permite que as ações do workflow, a ativação de ferramentas e o acesso, criação e atualização de seus itens sejam programados através de *scripts* (*VBScript*) dentro da ferramenta *Workflow Designer* e em páginas ASP que podem prover as intefaces de acessos do E2K.

3.8 Resumo do Capítulo

A aplicação de Sistemas de Gerência de *Workflow* como um componente na construção de Ambientes de Desenvolvimento de Software (ADS) é uma opção plausível, já que sistemas de *Workflow*, integrados a um ambiente de desenvolvimento, possibilitam o controle dos processos de software através da modelagem e execução destes processos. Como foi visto ao longo deste capítulo, os SGW atuais geralmente permitem sua utilização através da *Internet*, permitindo que estes funcionem sobre esta plataforma, fornecendo ao ambiente uma maior distribuição e portabilidade, já que o acesso pode ser feito via *navegador*.

No capítulo 5 uma arquitetura para ADS é proposta, integrando um SGW Comercial (*Microsoft Exchange 2000 Server*), algumas ferramentas comerciais de desenvolvimento (como editores da Rational [RAT 2001], compiladores e ambientes de programação - *Microsoft Visual Basic 6.0* e *Visual C++* [MIC 2001] e *Borland Delphi* e *JBuilder* [BOR 2002], etc.), e navegadores como interfaces de acesso; tudo isso funcionando sobre a plataforma *Internet*.

Ao longo da seção 3.6 (e suas sub-seções) alguns dos principais produtos de sistemas de gerência de *Workflow* foram descritos. Esta descrição deu ênfase as principais características, arquitetura dos produtos, conceitos associados a cada produto em particular e as formas de como os usuários acessam e utilizam estes produtos. Um maior detalhamento foi realizado em relação ao *Microsoft Exchange 2000 Server*, produto que foi utilizado no protótipo de experimentação deste trabalho. A seção 3.7 apresentou uma tabela comparativa entre os produtos de SGW apresentados.

4 Processo de Software

Processos de Software são entidades complexas que podem durar por longos períodos de tempo e são conduzidas através da interação de humanos e ferramentas computadorizadas [COR 93].

Um processo de software é um conjunto de passos parcialmente ordenados que pretendem alcançar uma meta; em engenharia de software a meta é construir um produto de software, ou atualizar um já existente [RAT 2001]. Outro conceito para processo de software é: um conjunto de fases de projeto, estágios, métodos, técnicas e práticas que pessoas empregam no desenvolvimento e manutenção do software e seus artefatos associados (planos, documentos, modelos, código, casos de teste, manuais, etc.) [AMB 2001].

Um processo de software descreve a interação entre pessoas e ferramentas na realização do trabalho envolvido no ciclo de vida do software. Um processo de software abrange o trabalho a ser realizado (as atividades), quem irá realizar (os desenvolvedores), o que será utilizado para realizar este trabalho (as ferramentas), o que será produzido (os artefatos) e quando e como este trabalho será realizado (o comportamento).

Durante as últimas décadas o problema de produção de software de alta qualidade tornou-se altamente complexo e difícil de gerir. Uma das razões é a rápida evolução das tecnologias e métodos de produção de software, juntamente com a complexidade das aplicações a serem desenvolvidas. Uma segunda razão é o caráter orientado a humanos dos processos de software, e as interações entre humanos e entre humanos e ferramentas que suportam suas atividades; que são altamente variáveis e imprevisíveis. Outra importante razão para a rápida evolução dos processos é a necessidade de (dinamicamente) adaptar o processo para acomodar os requisitos ou preferências de indivíduos que fazem parte do processo, ou para lidar com várias razões não previsíveis [COR 93].

Qualquer processo de software é composto por quatro elementos básicos: **Agentes, Papéis (*roles*), Artefatos** e **Atividades**. ***Agentes*** poderiam ser humanos ou sistemas de software, e estes possuem responsabilidades de efetuar uma **Atividade** atuando um determinado ***Papel***. ***Artefatos*** são produzidos a partir de um material não terminado ou outros produtos que são resultados de ***Atividades*** anteriores ou do processo todo. O trabalho dos ***Agentes*** poderia ser auxiliado por ferramentas [OCA 98].

Atividades são conduzidas por *agentes humanos* (possivelmente organizados em grupos ou times) auxiliados por ferramentas. Estes trabalham sobre componentes de software e documentos relacionados, chamados *artefatos de software* (ou itens de software). O conjunto dos artefatos constitui um *produto de software* [COR 93].

O processo de produção de software necessita ser modelado (explicitamente representado) no intuito de ser efetivamente repetível, automatizável e gerenciável. É preciso estar habilitado para controlar o processo e suportá-lo. Controlar o processo de

software se refere aos assuntos relacionados à habilidade de garantir que o processo evolui de acordo com regras e procedimentos específicos e desejáveis. Suportar o processo de software se refere à habilidade de automatizar os passos de produção, e guiar e solicitar as atividades conduzidas por humanos [COR 93]. Processo de produção de software é parte de um processo abrangente, chamado processo de software, o qual incluí *meta-atividades*, que suportam a evolução de todo o processo de software [COR 93].

4.1 Processos de Software

Muitos processos de software foram propostos e estão em uso. Nesta seção serão discutidos alguns destes processos, escolhidos devido ao caráter de orientação a objetos inerente a cada um dos processos e devido à sua grande aplicação [AMB 2001]:

- Processo ***OPEN*** (*Object-oriented Process, Environment and Notation*) – Processo, Ambiente e Notação Orientados a Objeto [OPE 2002] [AMB 2001];
- Processo ***OOSP*** (*Object-Oriented Software Process*) - Processo de Software Orientado a Objetos [PRO 2002] [AMB 2001] [AMB 98];
- Processo **RUP** (*Rational Unified Process*) - Processo Unificado Rational [RAT 2001] [RAT 98] [KRU 2000] [JAC 99].

4.1.1 Processo ***OPEN***

O processo *OPEN* foi desenvolvido pelo consórcio OPEN [OPE 2002], que é um grupo de indivíduos e organizações que promovem o uso da tecnologia orientada a objetos. Este processo pode ser utilizado por organizações que empregam a orientação a objetos e a utilização de componentes como tecnologias de desenvolvimento. *OPEN* utiliza as notações da UML [FOW 2000] [LAR 2000] [BOO 2000], OML (*Object Modeling Language*) ou qualquer outra notação orientada a objetos para documentar o software produzido pela aplicação do processo *OPEN* [AMB 2001] [OPE 2002].

O ciclo de vida dirigido por contrato (*contract-driven*) do *OPEN* é descrito na Figura 4.1. O lado esquerdo da Figura 4.1 representa as atividades para um único projeto, enquanto que as atividades do lado direito representam atividades que abrangem vários projetos. O ciclo de vida do *OPEN* inclui atividades que estão fora do escopo de um único projeto. Esta característica é chamada Gerenciamento de Programa (*programme management*) no processo *OPEN*, um programa como uma coleção de projetos e/ou *releases* de uma aplicação ou conjunto de aplicações [AMB 2001].

OPEN integra aspectos relacionados a negócios, qualidade, modelagem e reuso dentro do seu ciclo de vida que suporta o desenvolvimento de software utilizando o paradigma orientado a objetos [OPE 2002].

OPEN fornece flexibilidade. Seu *framework* baseado em metamodelo pode ser adaptado para domínios individuais ou projetos que levam em conta habilidades pessoais, cultura organizacional e requisitos peculiares para cada domínio industrial

[OPE 2002]. A seleção das tarefas e técnicas apropriadas para um projeto específico é parte do processo de adaptação do *OPEN* para os requisitos específicos de um projeto que são realizados na atividade "Iniciação do Projeto" [AMB 2001] (Figura 4.1).

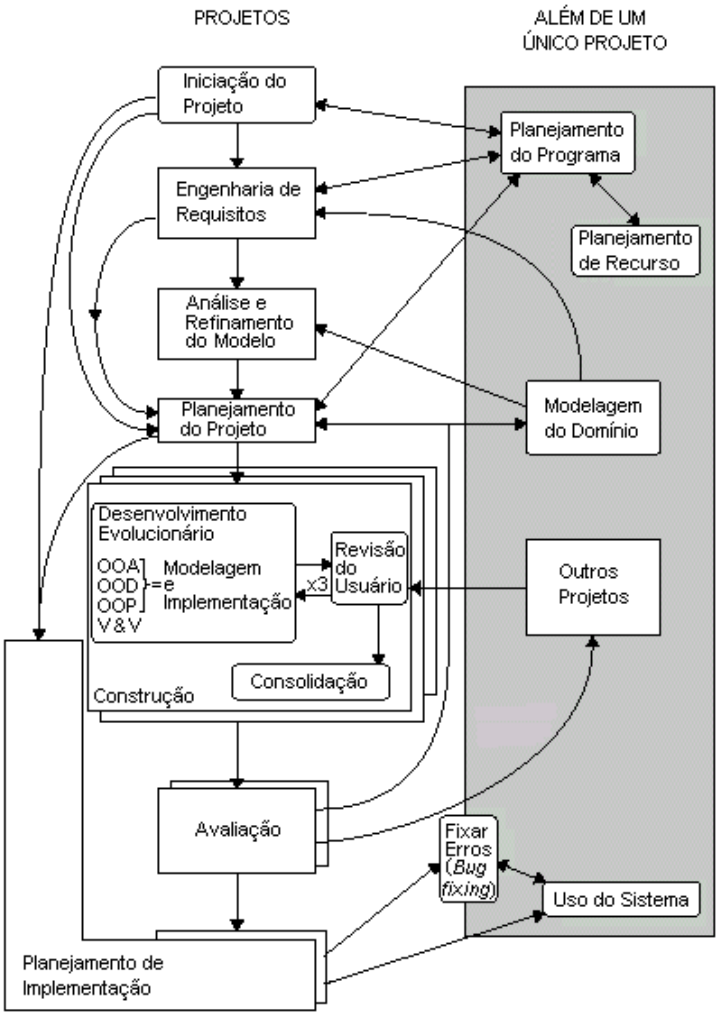

FIGURA 4.1 - Ciclo de Vida dirigido por contrato do processo OPEN [AMB 2001]

4.1.2 Processo ***OOSP***

O processo *OOSP* compreende uma coleção de padrões de processo. Um padrão de processo é uma coleção de técnicas gerais, ações, e/ou tarefas (atividades) que resolvem um problema específico de processo de software levando em conta determinados fatores. Assim como padrões de projeto [GAM 2000] descrevem soluções para problemas de projeto de software comuns, padrões de processo apresentam soluções para os problemas comuns que ocorrem com processos de software [AMB 2001] [AMB 98] [PRO 2002]. A Figura 4.2 descreve o ciclo de vida do processo OOSP.

OOSP é composto por quatro (4) fases de projeto – “Iniciar”, “Construir”, “Liberar” e “Manter e Suportar” (*Initiate*, *Construct*, *Deliver* e *Maintain and Support*) - sendo cada fase descrita por um padrão de processo “fase” correspondente. Existem também quatorze (14) estágios de projeto – Justificar, Definir e Validar os Requisitos

Iniciais, Definir os Documentos de Gerência Iniciais, Definir Infra-Estrutura, Modelar, Programar, Testar as Unidades, Generalizar, Testar o Sistema, Refazer, Liberar Versão, Avaliar, Suportar e Identificar Defeitos e Melhoramentos (*Justify, Define and Validate Initial Requirements, Define Initial Management Documents, Define Infrastructure, Model, Program, Test In The Small, Generalize, Test In The Large, Rework, Release, Assess, Support,* e *Identify Defects and Enhancements*) – cada um sendo descrito por um padrão de processo "estágio". Estágios de Processo são realizados de uma maneira iterativa dentro do escopo de um único projeto, já as fases de projeto são realizadas de uma maneira serial (seqüencial). Além das fases e estágios, *OOSP* também indica as importantes tarefas críticas (seta grande na parte inferior da Figura 4.2) para o sucesso do projeto que são aplicáveis para todos os estágios do desenvolvimento [AMB 2001] [AMB 98].

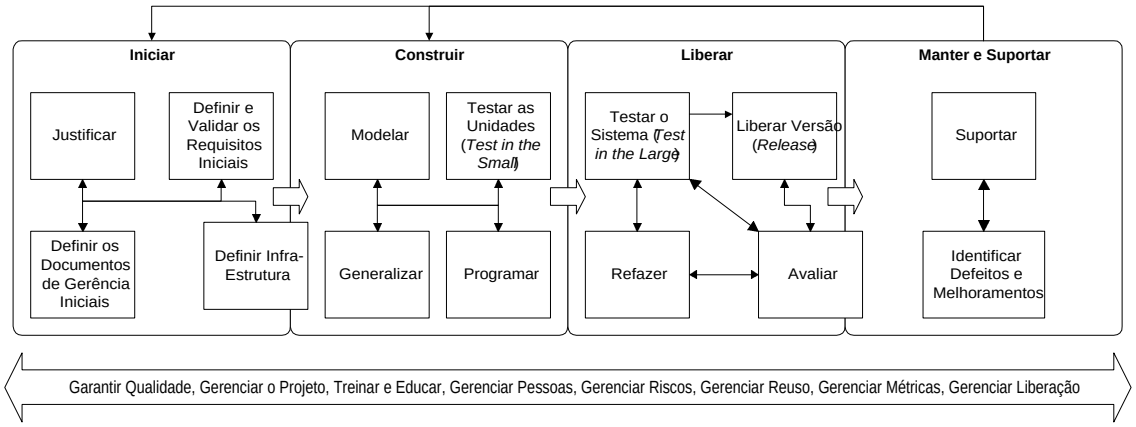

FIGURA 4.2 - Ciclo de Vida do Processo *OOSP* [AMB 2001]

4.1.3 Processo ***RUP***

O processo RUP é um processo de engenharia de software, fornecido através de uma base de conhecimento que pode ser consultada (serviços de busca) e disponibilizada com acesso através da *Internet* [RAT 98]. Este processo melhora a produtividade dos grupos de desenvolvimento e fornece as melhores práticas via manuais de orientação (*guidelines*), gabaritos (*templates*) e guias de ferramentas para todas atividades críticas do ciclo de vida de software. Este processo utiliza a notação UML [FOW 2000] [LAR 2000] [BOO 2000], que é um padrão industrial de notação orientada a objetos [RAT 98].

RUP é dividido em ciclos (ou iterações), cada ciclo trabalhando na geração de um executável interno que pode ser avaliado pela comunidade de usuários. Esta iteração diminui o risco do projeto, pois melhora a comunicação entre os desenvolvedores e os futuros usuários [AMB 2001]. Um ciclo de desenvolvimento do RUP divide-se em quatro fases consecutivas: **Concepção**, **Elaboração**, **Construção** e **Transição**. Cada fase é concluída com um marco (*milestone*) bem definido.

RUP é um processo de software configurável, podendo assim, ser adaptado de acordo com a realidade das empresas. RUP pode ser adaptado tanto para pequenos grupos de desenvolvimento até grandes organizações de desenvolvimento de software

[AMB 2001] [RAT 98]. A estrutura do RUP, contendo seus *Workflows* e fases, pode ser visualizada na Figura 4.3.

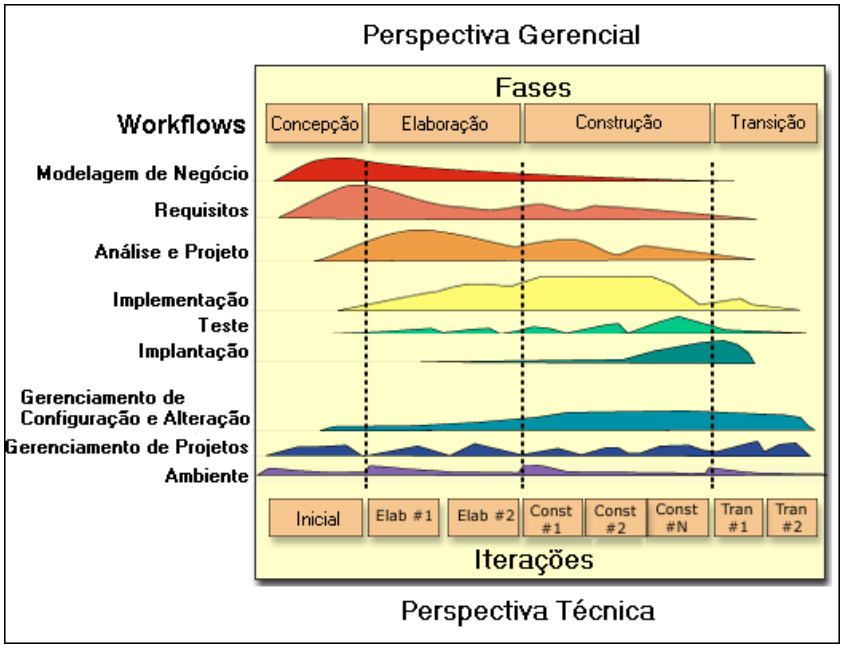

FIGURA 4.3 - RUP - Estrutura do Processo [RAT 98]

O processo **RUP** integra as melhores práticas de desenvolvimento de software moderno em uma forma que é conveniente para uma grande faixa de projetos e organizações. O emprego destas melhores práticas, através da aplicação do RUP como um guia, oferece aos grupos de desenvolvimento um número de vantagens chave. Estas melhores práticas são [RAT 98]:

(1) **Desenvolver o Software Iterativamente** - uma abordagem iterativa é necessária para permitir o crescente entendimento do problema através de sucessivos refinamentos e para incrementalmente encontrar a solução efetiva através de múltiplas iterações;

(2) **Gerenciar os Requisitos** - o RUP descreve como elicitar, organizar e documentar as funcionalidades e restrições necessárias; seguir e documentar acertos e decisões; e facilmente capturar e comunicar os requisitos de negócio;

(3) **Usar Arquiteturas baseadas em Componentes** - descreve como projetar uma arquitetura que é flexível, que suporta modificações, intuitivamente entendível e promove um reuso de software mais efetivo;

(4) **Visualmente Modelar o Software** - RUP mostra como modelar visualmente software para capturar a estrutura e o comportamento das arquiteturas e componentes;

(5) **Verificar a Qualidade de Software** - avaliação de qualidade é construída dentro do processo, em todas as atividades, envolvendo todos participantes, utilizando medidas e critérios objetivos, e não tratando como uma atividade posterior ou realizada por um grupo em separado;

(6) **Controlar as Modificações do Software** - RUP descreve como controlar, seguir e monitorar modificações para habilitar o desenvolvimento iterativo com sucesso.

O processo RUP foi empregado como processo do protótipo experimental desenvolvido durante este trabalho. Este processo foi avaliado e estudado no sentido de se extrair um processo simplificado, mas que abrangesse todo um ciclo de vida de desenvolvimento de software, e adaptado às ferramentas e tecnologias disponíveis para o desenvolvimento do protótipo.

O processo RUP foi escolhido por estar em bastante evidência tanto na comunidade acadêmica [AMB 2001] [MAN 2001] quanto industrial [COJ 99] [ERI 2000] [RAT 2001]. Muitas organizações de desenvolvimento de software já utilizam RUP como processo padrão em seus projetos; e alguns trabalhos acadêmicos exploram e estudam as características do RUP. Outro motivo é o fato do RUP integrar as melhores práticas de desenvolvimento de software [RAT 98], visando a construção de software de qualidade.

Na seção 4.2 o processo RUP é descrito em mais alguns aspectos.

4.2 Processo Unificado Rational - RUP

RUP é um processo de desenvolvimento de software desenvolvido pela *Rational* [RAT 2001]. RUP fornece todos os elementos necessários para o desenvolvimento de software, como informações sobre os *Workflows*, tipos de desenvolvedores, atividades, artefatos, manuais de orientação (*Guidelines*), ferramentas associadas, gabaritos (*templates*), etc. [RAT 2001]. Estes elementos fornecem aos usuários meios de conduzir, gerir e avaliar o processo de desenvolvimento de aplicações.

RUP é organizado por particionamento de *Workflow* das atividades (e trabalhadores relacionados) em grupos lógicos, áreas de preocupação, ou disciplinas [KRU 2000]. Um *Workflow* é uma seqüência de atividades que produzem um resultado de valor observável [RAT 2001]. No caso de processos de desenvolvimento de software, o resultado normalmente é representado por um, ou conjunto, de artefatos, como um documento, um programa ou um relatório de um teste ou uma revisão.

Estes grupos lógicos são divididos em seis *Workflows* de engenharia (ou processo), fornecendo uma **Perspectiva Técnica** que define a execução do processo através das quatro fases do ciclo de vida para desenvolver ou um novo sistema ou uma nova versão deste sistema (cada iteração produz uma versão do produto final - *releases*); e três *Workflows* de suporte, fornecendo uma **Perspectiva Gerencial** que possibilita a coordenação e gerência do processo no desenvolvimento iterativo de novas versões do sistema cada vez mais completas. Os *Workflows* de engenharia são: Modelagem de Negócios, Requisitos, Análise e Projeto, Implementação, Teste e Implantação. Os *Workflows* de suporte são: Gerenciamento de projeto, Gerenciamento de Configuração e Alteração e Ambiente [ERI 2000] [RAT 2001] [KRU 2000] [JAC 99] [RAT 98].

O Processo Unificado Rational (RUP) [RAT 2001] é um processo de engenharia de software e provê uma abordagem disciplinada para o assinalamento de tarefas e responsabilidades em uma organização de desenvolvimento [KRU 2000]. A

meta do RUP é garantir a produção de software de alta qualidade que cubra as necessidades dos usuários finais dentro de um cronograma e um orçamento previsíveis [JAC 99] [KRU 2000].

RUP apresenta uma abordagem bem definida sobre a gerência de projetos de software e processos de engenharia de software, mas não é uma abordagem centrada nas preocupações de sistemas de gerência. RUP não trata de assuntos como gerência de custos, gerência de recursos humanos, gerência de comunicação e gerência de aquisições [MAN 2001] [MAN 2001a].

O RUP é fornecido (fisicamente) como um conjunto de documentos HTML que trazem toda a informação necessária sobre os seus *Workflows*, fases, atividades, funções desempenhadas, gabaritos (*templates*), artefatos, manuais de orientação (*Work Guidelines*), etc. A Figura 4.4 mostra a interface de acesso ao menu principal das páginas do RUP. A partir deste menu é possível se obter qualquer informação sobre o RUP.

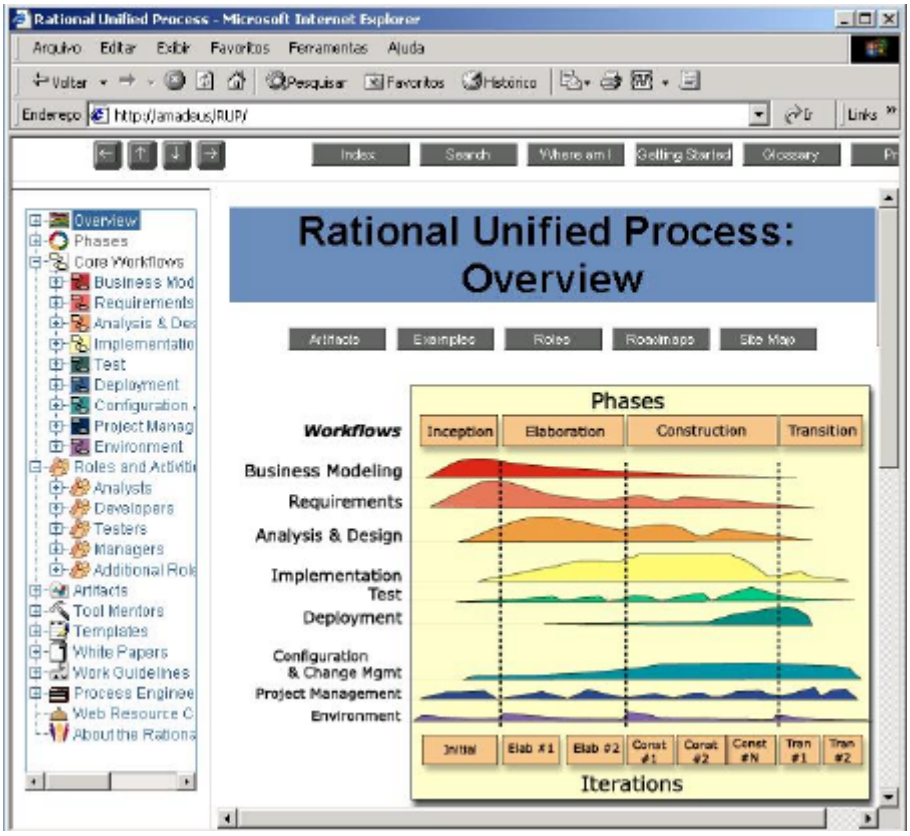

FIGURA 4.4 - Menu Principal do RUP

4.2.1 Alguns Conceitos Associados ao RUP

Um processo de software descreve "quem", agindo com determinado "comportamento", está fazendo "o quê", "como" e "quando". O RUP é representado utilizando alguns elementos de modelagem primários [RAT 98] [RAT 2001]:

- **Trabalhadores** (*Workers*) - "quem": Um trabalhador define o comportamento e responsabilidades de um indivíduo, ou um grupo de indivíduos trabalhando em um

time (grupo de trabalho). Pode-se pensar, por analogia, em um trabalhador usando um "chapéu" (*hat*) no projeto. Um indivíduo pode usar vários chapéus. Cada chapéu está relacionado a um papel (*role*) que pode ser desempenhado por um desenvolvedor, como por exemplo: Analista, Desenvolvedor, Testador, Gerente, etc.

- **Papel** (*Role*) - "Comportamento": papeis são tipicamente realizados por um indivíduo, ou por um grupo de indivíduos, que trabalham juntos como um time. Normalmente um membro de um grupo de projeto pode atuar em diferentes papeis. Papeis não são indivíduos, ao invés disso, estes descrevem como os indivíduos se comportam em um negócio (processo) e quais são as responsabilidades destes indivíduos (atuando um determinado papel). Um trabalhador realizando uma atividade atuando um determinado papel tem a responsabilidade de realizar um certo conjunto de atividades e produzir um certo conjunto de artefatos. Estas atividades e artefatos estão associados somente a um determinado papel.
- **Atividades** (*Activities*) - "como": Uma atividade de um trabalhador é uma unidade de trabalho que um indivíduo, atuando com um determinado chapéu (papel), deve realizar. Uma atividade tem um propósito claro, geralmente expressado em termos de criação e atualização de artefatos.
- **Artefatos** (*Artifacts*) - "o quê": Um artefato é um "pedaço de informação" (em forma de documentos, diagramas, códigos fonte, etc.) que é produzida, modificada ou utilizada por um processo. Artefatos são os produtos tangíveis do projeto, são os objetos que o projeto produz ou utiliza enquanto se trabalha na construção de um produto final.
- ***Workflows*** - "quando": Um *workflow* é uma seqüência de atividades que produzem um resultado de valor observável. Em termos de UML, um *workflow* pode ser expresso como um diagrama de seqüência, um diagrama de colaboração ou um diagrama de atividades.

A Figura 4.5 descreve graficamente os relacionamentos entre os conceitos de Atividade, Artefato, Papel e Trabalhador.

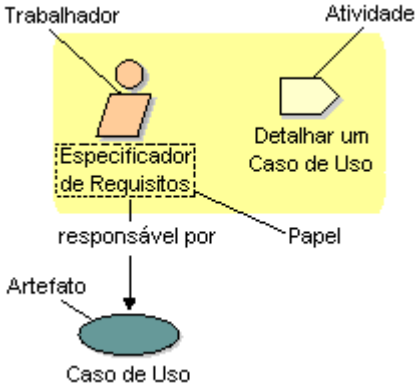

FIGURA 4.5 - Relacionamentos entre Atividade, Artefato, Papel e Trabalhador

4.2.2 *Workflows* e Artefatos Associados

RUP introduz seis *Workflows* de engenharia. Destes, cinco são diretamente ligados ao desenvolvimento de sistemas. Estes *Workflows* possuem um conjunto de artefatos associados aos mesmos. Este conjunto de artefatos pode ser o mesmo independentemente do domínio para o qual um sistema está sendo desenvolvido. Estes *Workflows* e artefatos serão descritos nesta seção [ERI 2000] [RAT 2001].

➢ **Modelagem de Negócios -** o propósito deste *Workflow* é a avaliação da organização na qual o sistema será utilizado. Esta avaliação visa o melhor entendimento das necessidades e problemas que devem ser resolvidos pelo sistema. Alguns artefatos associados a este *Workflow* são: modelo de casos de uso de negócio, modelo de objetos de negócio, glossário de negócios, documento de arquitetura de negócios, etc. A Figura 4.6 mostra o diagrama de atividades referente ao *Workflow* de Modelagem de Negócios.

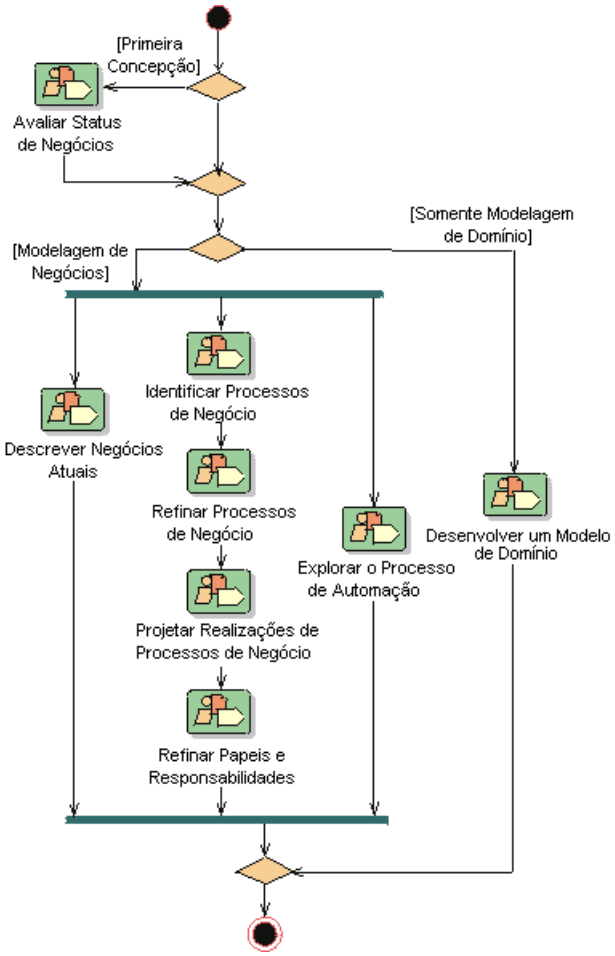

FIGURA 4.6 - Workflow de Modelagem de Negócios [RAT 2001]

➢ **Requisitos -** capturar e avaliar os requisitos dos interessados no produto (cliente e usuários finais), tendo um maior foco na usabilidade. Este *workflow* tem como objetivos: estabelecer entendimento com os interessados, definir escopo do projeto, estimar custos, etc. Um de seus principais artefatos é o modelo de casos de uso, com atores representando unidades externas se comunicando com o sistema, e casos de uso representando as seqüências de transação. Outros artefatos são: Glossário, Requisitos

dos Interessados, Documento Visão, Protótipo de Interface, etc. A Figura 4.7 mostra o *Workflow* de Requisitos.

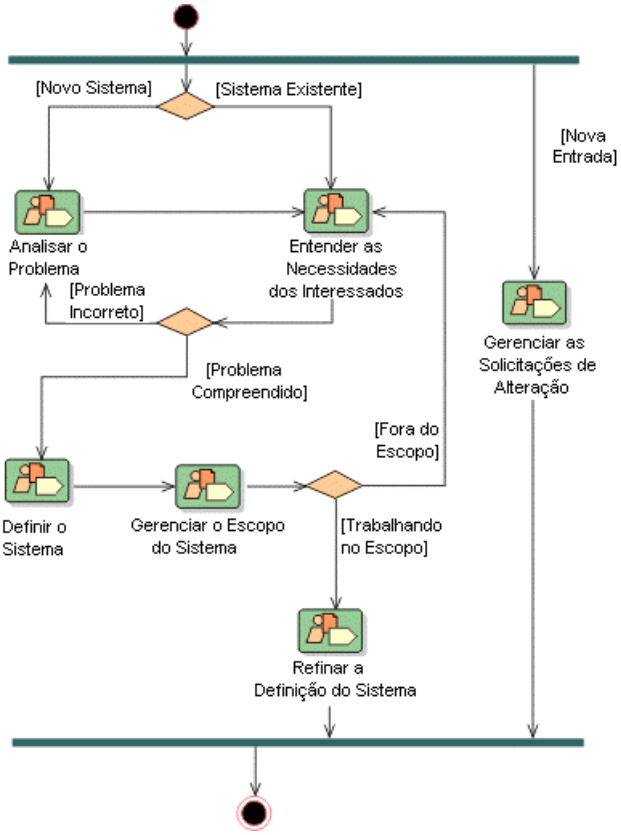

FIGURA 4.7 - Workflow de Requisitos [RAT 2001]

➢ **Análise e Projeto -** este *Workflow* visa investigar o ambiente de implementação pretendido e o efeito que este terá sobre a construção do sistema. Seus principais propósitos são transformar os requisitos em um produto de software, desenvolver uma arquitetura robusta para o sistema e adaptar o projeto para combinar com o ambiente de implementação, para fins de performance. Seu principal artefato é o modelo de objetos (modelo de projeto). Este pode incluir definições de interface para classes e subsistemas, especificando suas responsabilidades em termos de linguagem de implementação, distribuição, etc. Outros artefatos são: Documento de Arquitetura de Software, Modelo de Análise, Modelo de Implantação, Classes de Análise e Projeto, Modelo de Dados, etc. O diagrama de atividades referente ao *Workflow* de Análise e Projeto pode ser visualizado na Figura 4.8.

➢ **Implementação -** implementar o sistema no ambiente de implementação prescrito; este é o principal propósito deste *Workflow*, além de organizar os códigos fonte, testar os componentes desenvolvidos, integrar os resultados, etc. Este *Workflow* produz artefatos como: código-fonte, executáveis e arquivos de documentação. A Figura 4.9 mostra o diagrama de atividades do *Workflow* de Implementação.

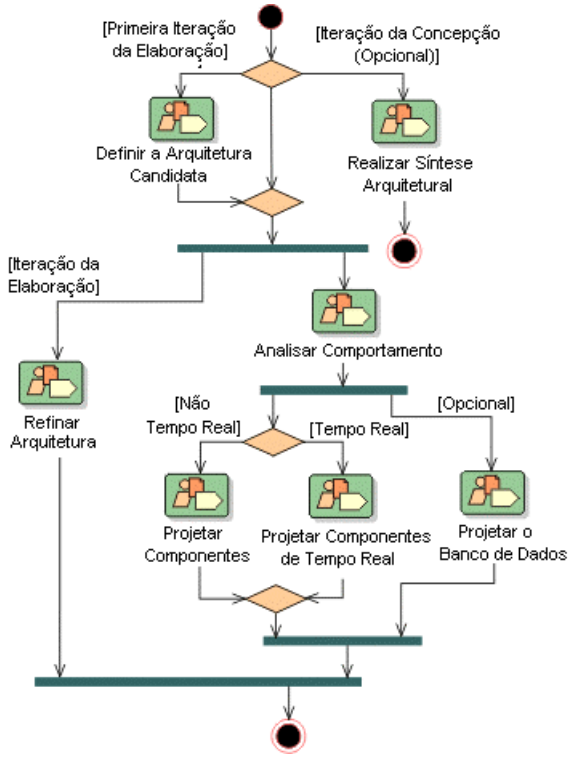

FIGURA 4.8 - Workflow de Análise e Projeto [RAT 2001]

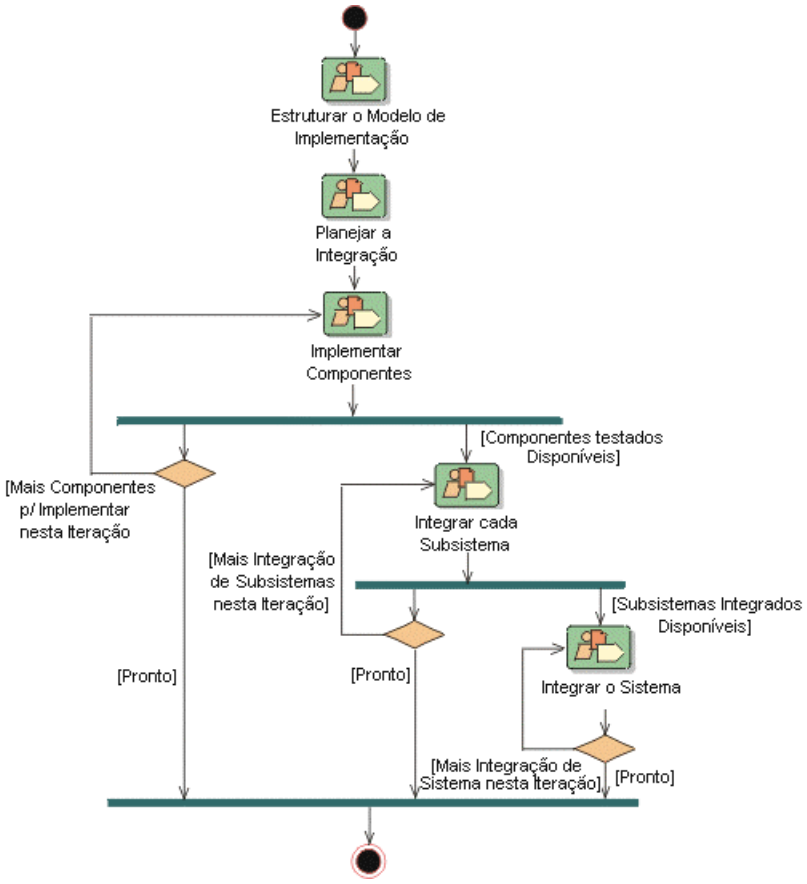

FIGURA 4.9 - Workflow de Implementação [RAT 2001]

➢ **Teste -** este *Workflow* tem o objetivo de garantir que o sistema final é aquele que foi solicitado pelos usuários, e que este não possua erros de implementação, através da verificação da iteração entre os objetos do sistema, integração entre os componentes, abrangência de todos os requisitos e identificação e correção dos defeitos encontrados. Os resultados (artefatos) deste *Workflow* são certificados que o sistema está pronto para uso em forma de resultados de testes. Além destes artefatos, cita-se: Plano de Teste, Procedimento de Teste, Caso de Teste, etc. A Figura 4.10 mostra o diagrama de atividades do *Workflow* de Teste.

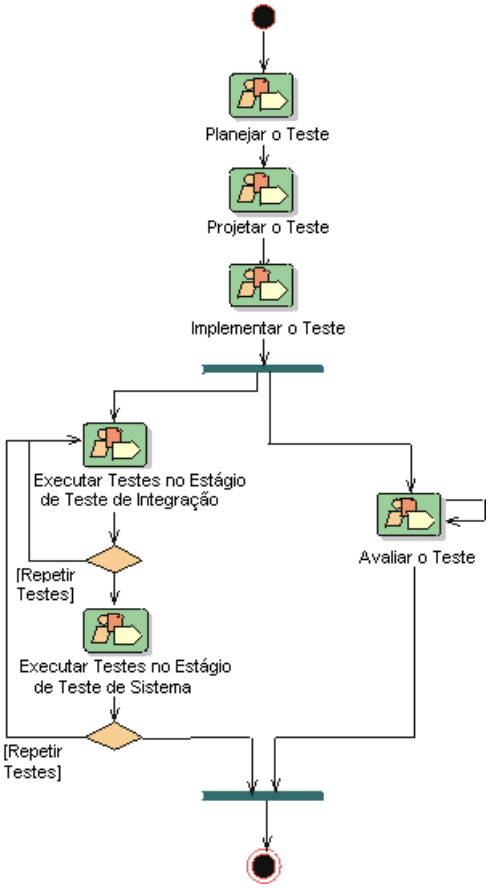

FIGURA 4.10 - Workflow de Teste [RAT 2001]

Os principais artefatos dos *workflows* descritos acima podem ser visualizados graficamente na Figura 4.11. Esta figura demonstra também as dependências de construção entre os artefatos, determinando quais artefatos já devem ter sido construídos (em atividades anteriores) para que uma determinada atividade possa ser iniciada.

Os demais *Workflows* do RUP também possuem artefatos associados, mas que não estão diretamente ligados ao desenvolvimento do sistema em si, e sim na gerência dos projetos, fornecimento de um ambiente de desenvolvimento adaptado ao sistema em desenvolvimento, gerenciamento de alterações e adaptações que podem ocorrer durante um projeto, disponibilização dos produtos, enfim, trata da gerência dos projetos, do ambiente e dos elementos associados ao produto em construção. Estes *Workflows* são citados abaixo, juntamente com alguns artefatos associados.

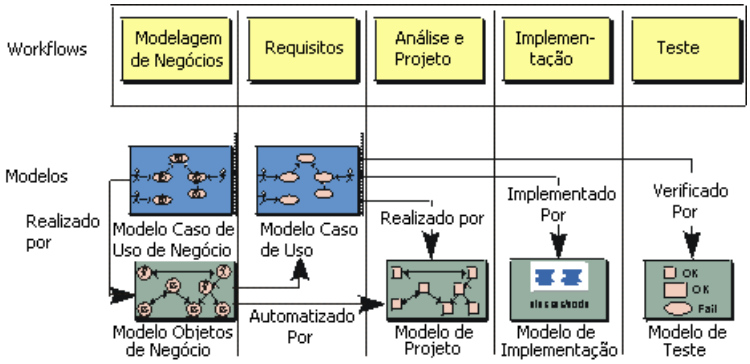

FIGURA 4.11 - *Workflows* e Artefatos Associados [ERI 2000]

➢ **Implantação**: o *Workflow* de implantação descreve as atividades que garantem que o produto de software produzido está disponível para o uso. Alguns artefatos associados a este *Workflow* são: fatura de materiais, *release notes*, o próprio produto, artefatos de instalação, material de treinamento, manual do usuário, etc. O *Workflow* de Implantação é descrito na Figura 4.12.

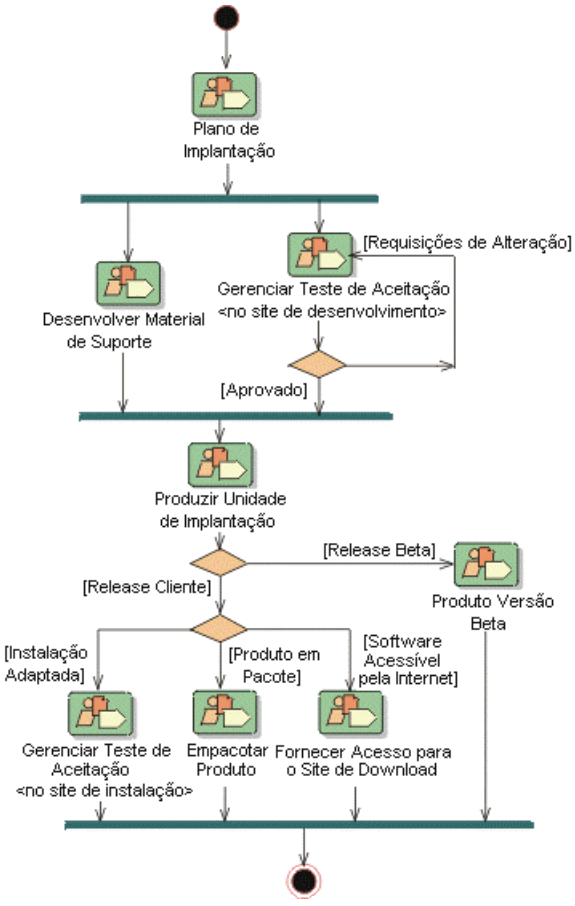

FIGURA 4.12 - Workflow de Implantação [RAT 2001]

➢ **Gerenciamento de Configuração e Alteração**: este *Workflow* controla as alterações e mantém a integridade dos artefatos do projeto. Envolve a identificação de itens de configuração, restrição de alteração, auditoria das modificações e define e gerência configurações dos itens. Como artefatos associados a este *Workflow* pode se citar: Plano de Gerência de Configuração e Requisição de Alteração (*Change Request*). A Figura 4.13 demonstra o *Workflow* de Gerenciamento de Configuração e Alteração.

➢ **Gerenciamento de Projeto**: procura balancear os objetivos concorrentes, gerenciar os riscos e superar as restrições para liberar, com sucesso, um produto de software que encontre as necessidades dos clientes e usuários. Seu propósito é: (a) fornecer um *framework* para gerência de projetos de software; (b) fornecer manuais de orientação (*guidelines*) para planejamento, pessoal, execução e monitoramento dos projetos e; (c) fornecer um *framework* para gerência de riscos. Alguns de seus artefatos são: Plano de Desenvolvimento de Software, Caso de Negócio, Plano de Iteração, Registro de Revisões (armazena resultados de revisões), Plano de Garantia de Qualidade, etc. O *Workflow* de Gerenciamento de Projeto é mostrado na Figura 4.14.

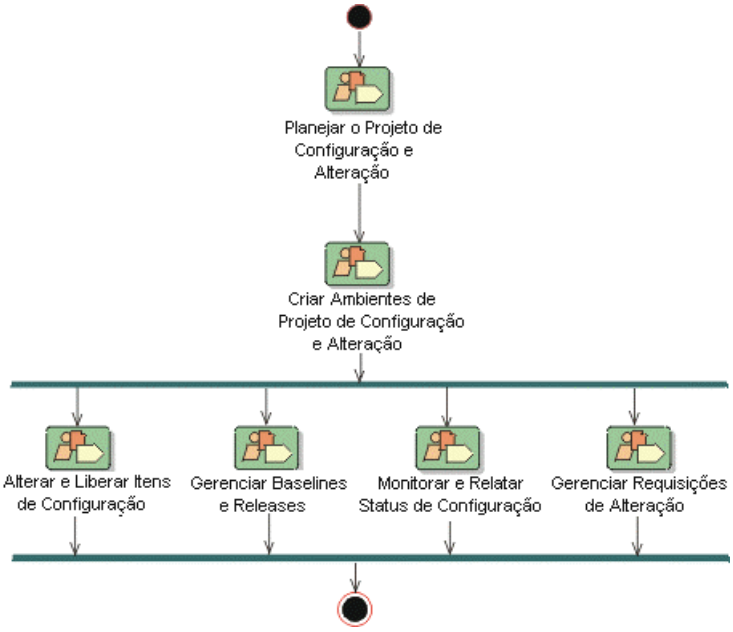

FIGURA 4.13 - Workflow de Gerenciamento de Configuração e Alteração [RAT 2001]

➢ **Ambiente**: este workflow tem seu foco sobre as atividades necessárias para configurar o processo para um projeto. O propósito das atividades deste Workflow é fornecer à organização de desenvolvimento de software o ambiente de desenvolvimento de software – ambos processos e ferramentas – que irão suportar o grupo de desenvolvimento. Alguns artefatos são: Caso de desenvolvimento, *templates* específicos de projeto, ferramentas, manuais de orientação em geral, etc. O *Workflow* de Ambiente é descrito graficamente através do diagrama de atividades da Figura 4.15.

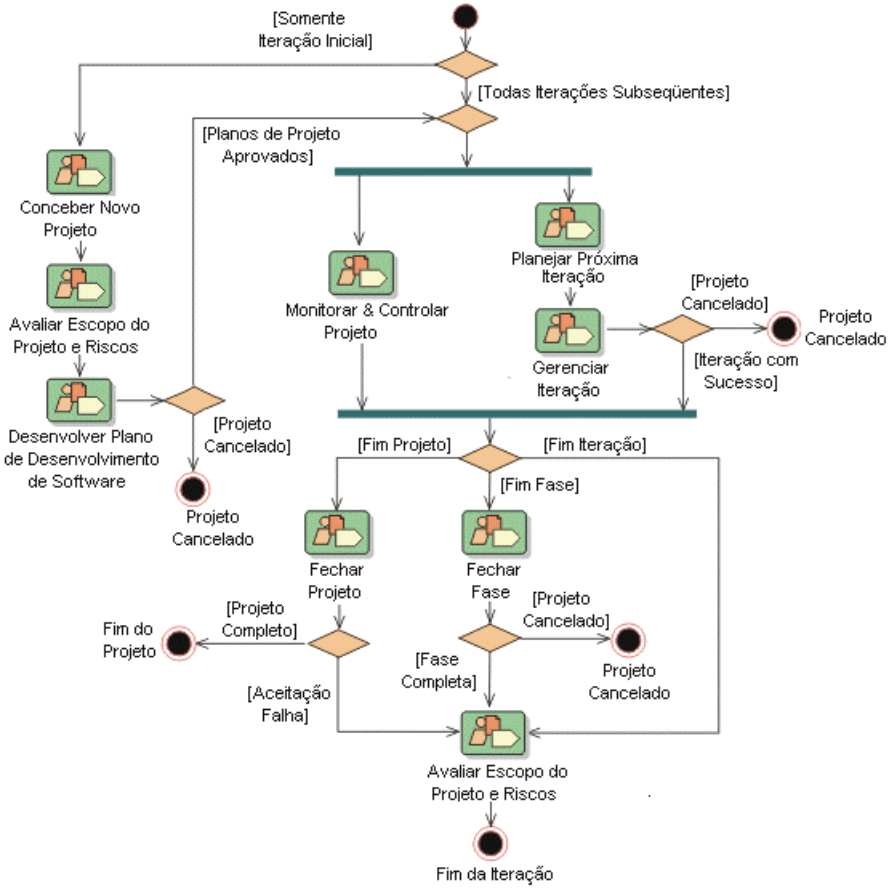

FIGURA 4.14 - Workflow de Gerenciamento de Projeto [RAT 2001]

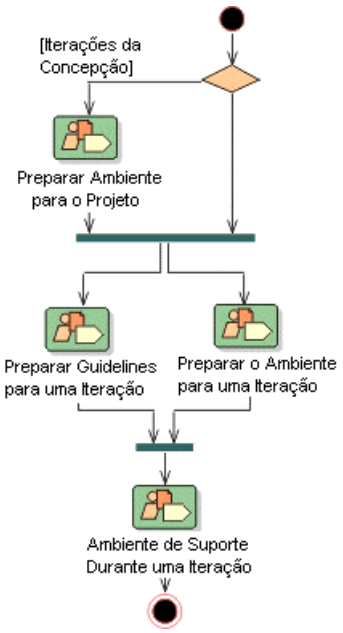

FIGURA 4.15 - Workflow de Ambiente [RAT 2001]

4.3 Requisitos para um Processo de Software

A qualidade de software é largamente determinada pela qualidade dos processos utilizados para seu desenvolvimento. Desde modo, a melhoria da qualidade de software é obtida pela melhoria da qualidade dos processos [MAN 2001]. Definição de processo é reconhecido como um elemento crítico no melhoramento de processo de software [CHR 99]. Isto é refletido no modelo CMM (*Capability Maturity Model*), proposto pelo SEI (*Software Engineering Institute*) [SEI 2002], onde é defendido que a definição de processo de software é fundamental para alcançar altos níveis de maturidade no desenvolvimento de software. O modelo CMM é um *framework* a partir do qual um processo para grandes e complexos esforços de desenvolvimento de software pode ser definido [PAU 93] [SEI 2002].

Um *processo de software*, segundo o modelo CMM, pode ser classificado segundo cinco níveis de maturidade, descritos na Tabela 4.1. Cada nível de maturidade possui algumas "Áreas-Chave de Processo" (*KPA – Key Process Areas*) e um processo de software deve satisfazer todas as áreas-chave definidas em um nível do CMM para alcançar este nível [AMB 2001].

TABELA 4.1 – Os Cinco Níveis de Maturidade do Modelo CMM [AMB 2001]

Nível	Descrição	Características
1. Inicial	O processo de software é *ad hoc*, e ocasionalmente caótico. Alguns processos são definidos e o sucesso depende de esforços individuais e heroícos.	- Supercomprometimento é comum - Em momentos de crise, procedimentos planejados são abandonados e grupos são alocados para codificação e teste - Sucesso depende do gerente e de um grupo efetivo - Processo de software é desconhecido. Recursos são utilizados e possivelmente um produto de software será o resultado
2. Repetível	Processos de Gerência de Projetos Básicos são estabelecidos para gerenciar custos, cronogramas e funcionalidade. A disciplina de processo necessária está pronta para repetir os sucessos anteriores de projetos similares.	- Planejemento e Gerência de novos projetos são baseados em experiências anteriores com projetos similares - Técnicas de gerência de processo melhoram a capacidade do processo - Diferenças entre projetos tendem a reduzir o reuso e trabalho em grupo - A comunidade de usuários pode visualizar o andamento do projeto em poucas ocasiões – como revisões – permitindo limitado controle de gerenciamento
3. Definido	O processo de software para ambas atividades de gerência e desenvolvimento é documentado, padronizado e integrado em um processo de software padrão para toda a organização. Todos projetos utilizam uma versão aprovada do processo de software padrão da organização.	- Um processo padrão é utilizado - Gerência possui uma boa visualização do progresso técnico no projeto - Processos definidos permitem aos usuários uma maior visibilidade do projeto e possibilita atualizações precisas e rápidas do estado do projeto.
4. Gerenciado	Medidas detalhadas, chamadas métricas, da qualidade do processo de software e do produto são coletadas. Ambos processo de software e produto são quantitativamente entendidos e controlados.	- Produtividade e qualidade são quantificadas (medidas) por meio de importantes atividades do processo de software em todos os projetos - Os usuários podem estabelecer um entendimento preciso e quantitativo da capacidade do processo de software e risco do projeto antes do mesmo começar
5. Otimizado	Melhoramento contínuo do processo é habilitado pela realimentação quantitativa do processo de software e das idéias inovativas e tecnologias.	- Inovações que exploram as melhores práticas são identificadas e compartilhadas com toda organização - O processo de software é melhorado pela modificação das causa comuns de ineficiência - Modificações disciplinadas é a norma, não a exceção - A comunidade de usuários e a organização de software trabalham em conjunto para estabelecer um relacionamento de sucesso e forte

Na Tabela 4.2 pode ser visualizado um quadro comparativo entre os três processos descritos nesta seção. Este quadro foi extraído a partir dos trabalhos de Scott W. Ambler nas organizações EUP - *Enterprise Unified Process* e *Ronin International Inc.* [AMB 2001] [EUP 2003] [RON 2003] – e de Lisandra Manzoni no Grupo AMADEUS [AMA 2002] [MAN 2001] [MAN 2001a], e faz a avaliação dos processos em relação as áreas-chave de processo do modelo CMM. Cada processo tem um conceito relativo ao suporte, ou não, das Áreas-Chave de Processo do CMM.

TABELA 4.2 – Comparativo dos Processos RUP, OPEN e OOSP em relação as Áreas-Chave de Processo do Modelo CMM [AMB 2001]

Áreas-Chave de Processo (Nível CMM)	**RUP**	**OPEN**	**OOSP**
Prevenção de Defeitos (5) – analisa defeitos que foram encontrados no passado e toma as devidas ações para prevenção destes defeitos	Bom	Muito Bom	Muito Bom
Gerência de Software Integrada (3) – integra as atividades de Engenharia de Software e de Gerência em um processo definido e coerente para cada projeto.	Muito Bom	Muito Bom	Muito Bom
Coordenação Intergrupos (3) – participa com outros grupos de toda a organização para lidar com requisitos, objetivos e assuntos ligados a organização.	Bom	Muito Bom	Bom
Definição dos Processos da Organização (3) – desenvolve e mantém o processo da organização (descrições, *guidelines*, critérios, BD, documentação, etc.).	Muito Bom	Muito Bom	Muito Bom
Foco nos Processos da Organização (3) – desenvolve e mantém um entendimento da organização e dos processos de software dos projetos; e coordena as atividades de avaliação, desenvolvimento, manutenção e melhora dos processos.	Muito Bom	Bom	Muito Bom
Revisão Conjunta (ou Revisão por Pares) (3) – exame metódico dos projetos liberados e identificação de defeitos e pontos a serem melhorados.	Muito Bom	Ótimo	Ótimo
Gerência de Modificação de Processo (5) – define metas de melhoramento do processo e proativa e sistematicamente identifica, avalia e implementa melhoramentos no processo de software da organização.	Bom	Bom	Muito Bom
Gerência Quantitativa do Processo (4) – estabela objetivos e, após, mede a performance de um processo definido para o projeto.	Muito Bom	Muito Bom	Bom
Gerência de Requisitos (2) – estabelece, documenta e mantém um acordo com a comunidade de usuários baseado nos requisitos do projeto.	Muito Bom	Ótimo	Ótimo
Gerência de Configuração de Software (2) – estabelece e mantém a integridade das versões liberadas pelo projeto durante todo o ciclo de vida.	Muito Bom	Muito Bom	Muito Bom
Engenharia de Produto de Software (3) – realiza as tarefas de engenharia para construir e manter o software de acordo com o processo de software definido para o projeto e ferramentas e métodos apropriados.	Ótimo	Ótimo	Ótimo
Planejamento de Projetos de Software (2) – desenvolve e negocia estimativas para o término do trabalho, estabelecendo compromissos e definindo o plano de trabalho.	Ótimo	Muito Bom	Muito Bom
Acompanhamento e Supervisão de Projetos de Software (2) – fornece visibilidade para o progresso atual de um projeto para que a gerência possa tomar ações efetivas e apropriadas.	Muito Bom	Ótimo	Muito Bom
Garantia de Qualidade de Software (2) – faz revisões e auditorias sobre as versões liberadas pelo projeto e verifica se as mesmas estão de acordo com os padrões aplicáveis, *guidelines* e processos.	Muito Bom	Ótimo	Ótimo
Gerência de Qualidade de Software (4) – define metas de qualidade para produtos de software e estabelece os planos para alcançar estas metas.	Muito Bom	Ótimo	Ótimo
Gerência de Subcontratação (2) – seleciona e efetivamente faz a gerência de subcontratados no desenvolvimento de software de qualidade.	Razoável	Muito Bom	Muito Bom
Gerência de Modificação de Tecnologia (5) – identifica, seleciona e avalia novas tecnologias e incorpora as mesmas de maneira efetiva à organização.	Razoável	Muito Bom	Muito Bom
Programa de Treinamento (3) – identifica o treinamento necessário à organização, projetos e indivíduos e então desenvolve o treinamento correspondente a estas necessidades.	Razoável	Muito Bom	Ótimo

4.4 Limitações do RUP

A avaliação do RUP baseado no modelo CMM mostra que alguns dos aspectos do processo de software são pouco suportados, ou mesmo não suportados, pelo RUP.

O Processo Unificado apresenta uma abordagem bem-definida em gerência de projetos e processos de engenharia de software, mas sua abordagem não se concentra em atividades de gerência de sistemas. RUP apresenta lacunas em atividades envolvendo gerência de recursos humanos, gerência de custos e gerência de aquisição [MAN 2001].

Atividades relacionadas à gerência de subcontratação e programa de treinamento não são suportadas pelo RUP. RUP também não descreve atividades para melhorar o processo de software da organização, apenas descreve uma avaliação do processo, mas não deixa claro como os resultados da avaliação do processo da organização alimentam a melhoria contínua do processo [MAN 2001] [MAN 2001a].

As áreas-chave do CMM melhor suportadas pelo RUP pelo são Gerência de Configuração de Software, Gerência de Requisitos, Planejamento de Projetos de Software, Acompanhamento e Supervisão de Projetos de Software, Definição dos Processos da Organização, Gerência de Qualidade de Software, Gerência Quantitativa do Processo e Engenharia de Produto de Software [MAN 2001] [AMB 2001].

RUP oferece um bom suporte para as áreas-chave de Gerência de Software Integrada e Coordenação Intergrupos. RUP oferece pouco suporte para as áreas-chave de Garantia de Qualidade de Software, Foco nos Processos da Organização, Gerência de Modificação de Processo, Prevenção de Defeitos, Gerência de Modificação de Tecnologia e Revisão por Pares [MAN 2001] [AMB 2001] [AMB 2001].

4.5 Aplicação de SGW na Modelagem e Execução de Processos de Software

Desenvolvimento de software pode ser considerado como um tipo especializado de processo de negócio, de maneira que documentos (especificações, definições arquiteturais, código fonte, diagramas, etc.), informação (resultados de testes, análises de revisões, solicitações de modificação, cronogramas, anotações, etc.) e tarefas (elicitações de requisitos, colaboração no projeto e codificação, revisões, etc.) são passadas de um participante (projetistas, analistas, especialistas, programadores, testadores, avaliadores, revisores, gerentes, etc.) para outro de acordo com um conjunto de regras conhecidas como o processo de desenvolvimento [BAR 2000] e onde tarefas gerenciais de alocação de atividades aos desenvolvedores, planejamento de prazos e contratação de pessoas é realizada pela gerência dos projetos [CHA 97] [BUS 98].

Se um processo de software é um tipo especializado de processo de negócio e processo de negócio pode ser automatizado por SGW (Capítulo 3), então processos de software também podem ser automatizados por estes sistemas de software. Como já mencionado neste capítulo, SGW fazem a automação de processos de negócio. O termo *processos de negócio* pode ser muito genérico, se for considerado que o processo de negócio de uma organização de desenvolvimento de software é *fazer* produtos de

software; desta maneira, pode-se dizer que processo de software é um subconjunto, ou está inserido dentro da gerência de workflow (SGW) [OCA 98].

Alguns autores já defenderam o uso do paradigma de *Workflow* como a base para suportar a execução e o gerenciamento de processos de desenvolvimento de software [ARA 99] [BAR 2000] [BAR 2000a] [CHA 97] [CHA 97a] [CHA 99] [GRU 2000] [OCA 98] [MAU 99] [MAN 2001]. Um SGW pode ser utilizado no suporte a processo de software porque o mesmo fornece abertura (*openness*) para interoperar com outros sistemas, um ambiente de execução distribuída, facilidades de monitoramento, gerência de recursos humanos e flexibilidade à mudanças [CHA 97].

A aplicação de SGW em Ambientes de Desenvolvimento de Software como um "gerenciador de processos" parece ser uma boa idéia, visto que a construção de ambientes com características como gerência do processo, integração de ferramentas, assinalamento de atividades e monitoramento de projetos; é muito complexa se feita por completo. Assim, a utilização de SGW pode facilitar a construção de Ambientes de Engenharia de Software que provêm estas características.

A seguir algumas das características associadas à aplicação de SGW na modelagem e execução de processos de software serão ressaltadas.

- **Modelagem de Processos**: SGW oferecem linguagens de programação de alto nível para a definição de processos, em sua maioria de forma gráfica, facilitando bastante a definição de modelos de processo. Estes modelos de processo podem ser executados no “motor de *workflow*” (*workflow engine*) [ARA 99].
- **Separação entre o modelo de processo e suas instâncias em execução (Evolução do Processo)**: o principal propósito de uma ferramenta de *workflow* é permitir que a lógica do processo possa ser modificada separadamente da lógica da atividade embutida nas aplicações dos usuários. Com esta funcionalidade, processos de software podem ter suas definições alteradas, sem afetar as instâncias de processo em execução, ou até refletindo as alterações diretamente nestas instâncias, que passam a seguir a nova definição [ARA 99] [OCA 98].
- **Execução do Processo**: a execução de processos de software em SGW corresponde à instanciação das atividades definidas no modelo de processo. Esta instanciação compreende a atualização das listas de trabalho da cada participante do processo com a descrição das tarefas que devem realizar [ARA 99].
- **Cooperação**: SGW são considerados ferramentas cooperativas, uma vez que coordenam as atividades de grupos de trabalho. Poucas são as propostas em ADS que tratam das interações cooperativas em projetos de software. SGW demonstram uma maior preocupação com os aspectos de cooperação em processos de trabalho, principalmente na oferta de recursos para a localização dos participantes do processo no contexto de sua realização, na negociação de papéis e tarefas e na oferta de canais de comunicação mais diretos entre os executores do processo [ARA 99]. Nem todas abordagens desenvolvidas em SGW e ambientes de suporte a processos de software (ADS-CP) fornecem um *framework* conveniente para a cooperação, e se suportam esta, não abrangem todos os tópicos de cooperação: coordenação, comunicação e colaboração [OCA 98].
- **Monitoramento**: SGW provêem características que oferecem facilidades de coordenação, monitoramento e auditoria de processo. Oferecem diversas

informações sobre o andamento do processo. Estas características podem ser um recurso inicial para a coleta de métricas sobre a realização dos processos e a indicação de pontos de melhoramento [ARA 99].

- **Controle**: SGW têm sido desenvolvidos fornecendo maior controle sobre humanos do que outros sistemas, possibilitando, desta maneira, melhores condições para a cooperação. Processos reais possuem características de cooperação e controle e, dependendo do tipo de negócios, os processos poderiam necessitar de controle ou cooperação. Processo de Software possui ambas características de cooperação e controle. Desenvolvimento de software é um processo que envolve agentes cooperando para produzir programas, mas também algum grau de controle deve existir. Este grau depende de quão maduro é o processo e a organização que o aplica [OCA 98].
- **Papel humano no gerenciamento de processo**: este é um dos fatores que diferenciam as tradicionais aplicações de SGW e as aplicações de processo de software (ADS-CP). Normalmente, aplicações de SGW têm algumas atividades executadas por agentes automáticos (software) e outras por humanos. Por outro lado, em aplicações de processos de software, não existem atividades (ou mesmo, macro atividades) que sejam conduzidas inteiramente por aplicações de software, embora algumas das tarefas (pode-se considerar atividade como uma composição de várias tarefas, ou, macro atividade como uma composição de várias atividades) podem ser completamente automáticas, por exemplo, compilar um arquivo [OCA 98].
- **Heterogeneidade**: SGW são sistemas projetados para operar em ambientes de informação modernos (distribuídos, cliente/servidor, e heterogêneos). A possibilidade de operar em ambientes de informação distribuídos e heterogêneos é uma característica interessante quando se pensa em projetos de construção de software multidisciplinares, envolvendo diversos setores de uma organização ou mesmo organizações diferentes com agentes espalhados em locais distantes entre si. Além disso, fabricantes de SGW possuem a preocupação com a integração de seus produtos com outros SGW, permitindo que processos possam ser executados por serviços de *workflow* distintos, integrados através de uma interface pré-definida [ARA 99].
- **Integração de ferramentas**: aplicações novas ou já existentes podem ser facilmente incorporadas nos ambientes sem grandes modificações tanto no sistema de controle como na aplicação sendo integrada. De acordo com o modelo de referência para *workflow* (Figura 3.1) [HOL 95], as infra-estruturas de aplicações de *workflow* são mais maleáveis, permitindo a integração em maior escala de ferramentas de apoio, o que ainda não é trivial em ADS [ARA 99].

4.6 Resumo do Capítulo

A gerência dos Processos de Software é um tópico de pesquisa importante na área de Engenharia de Software [ARA 99] [ARC 2001] [BAR 2000] [BAR 2000a] [CAL 96] [CHA 97] [CHA 97a] [COR 93] [CHA 99] [EPO 2001] [FIN 94] [GAR 94]

[GIM 99] [OCA 98] [MAU 99] [MAN 2001] e vários modelos de processos de software já foram propostos (por exemplo: *OPEN*, *OOSP* e *RUP*).

O Processo Unificado Rational (RUP) é um modelo de processo que vem, cada vez mais, sendo aplicado pelas empresas de desenvolvimento como o processo de software a ser seguido durante a construção de um novo produto de software. Além de fornecer todos os *Workflows*, RUP também fornece os artefatos e gabaritos (*templates*) associados a estes artefatos; os papéis, ou funções (*Roles*), associados às atividades dos *Workflows* que devem ser desenvolvidas por indivíduos que desempenham estas funções; as ferramentas que podem ser utilizadas para a execução das atividades; os manuais de orientação (*work guidelines*) para execução das atividades; textos com conceitos e informações detalhadas a respeito de vários aspectos sobre a aplicação do RUP; alguns estudos de caso (em forma de *White Papers*) mostrando a aplicação do RUP em projetos anteriores; entre outros. RUP, segundo seus autores, também integra as melhores práticas de desenvolvimento de software e é o processo em evidência na atualidade, tanto na área acadêmica quanto industrial. Com todo este aparato, RUP fornece meios para a produção disciplinada e organizada de software e pode ser aplicado, através de uma adaptação do mesmo, como o processo de software a ser modelado e executado no SGW, que por sua vez está integrado ao protótipo de experimentação desenvolvido durante este trabalho.

A aplicação de SGW para a gerência dos processos dentro de ADS ainda é um tópico de pesquisa interessante [ARA 99] [BAR 2000] [BAR 2000a] [CHA 97] [CHA 97a] [CHA 99] [OCA 98] [MAU 99] [MAN 2001], visto que SGW fornecem características interessantes como interoperabilidade, ambiente de execução distribuída, facilidades de monitoramento, gerência de recursos humanos e flexibilidade à mudanças; e ainda pode ser utilizado como base para a construção de novos ADS que podem ser construídos sobre SGW e integração de outras ferramentas comerciais de desenvolvimento de software.

5 Arquitetura Baseada em SGW e Ferramentas Comerciais para Construção de ADS

Este capítulo apresenta o problema de construção de Ambientes de Desenvolvimento de Software (ADS) a partir da integração e aplicação de ferramentas comerciais (COTS) como componentes de construção, mostrando também alguns trabalhos relacionados a este tema. Na seção 5.4 é apresentada uma arquitetura para a construção de ambientes de desenvolvimento de software. A seção 5.5 apresenta as vantagens e desvantagens inerentes à abordagem de construção apresentada na seção 5.4. Em seguida é feita uma comparação da arquitetura apresentada com alguns trabalhos relacionados (Seção 5.6) e após (Seção 5.7) a arquitetura proposta na seção 5.4 é avaliada de acordo com os requisitos descritos na seção 2.4. Neste capítulo, além de ADS, também será utilizada a sigla ADS-CP (Ambientes de Desenvolvimento de Software Centrados no Processo), visto que a arquitetura apresentada também visa a gerência dos processos de software. Poderá ser utilizado somente o termo ADS, mas também com a intenção de incluir a gerência dos processos de software no ambiente.

5.1 Uso de Produtos Comerciais de Prateleira (COTS) como Componentes de Construção

O mundo do desenvolvimento de software tem evoluído rapidamente na última década. Em particular, o uso de produtos de prateleira (*commercial off-the-shelf* - COTS; produtos comerciais de prateleira) como elementos de grandes sistemas está se tornando uma prática comum, devido à redução de orçamentos, aceleração da taxa de melhoramento dos produtos COTS e expansão dos requisitos dos sistemas [MOR 2000].

O termo COTS é muito genérico. Este pode se referir para muitos tipos e níveis diferentes de software, por exemplo: software que fornece uma funcionalidade específica ou uma ferramenta usada para gerar código. O termo COTS significa um produto de software, fornecido por um vendedor, que possui uma, ou mesmo várias, funcionalidades fazendo parte de um sistema. É um pedaço pré-construído de software que é integrado em um sistema e deve ser liberado com este sistema para fornecer uma funcionalidade operacional ou para manter esforços de manutenção [MOR 2000].

Como o tamanho e complexidade dos sistemas continuam a crescer, o uso de componentes COTS está sendo visto cada vez mais como uma solução promissora e, também, talvez inevitável para este problema [CHU 2002]. Outro argumento a favor do uso ou reuso de software COTS envolve a redução do risco de desenvolvimento [LOO 99].

Produtos COTS são normalmente bem testados e podem fornecer um sistema confiável que suporta as necessidades dos usuários. Desenvolvimentos recentes em engenharia de software incluem o uso de produtos COTS com sistemas de software

adaptados ou pré-existentes para melhorar a funcionalidade. Entretanto, integrar um sistema existente com um produto COTS têm se mostrado, em algumas vezes, uma tarefa complicada (necessitando de um considerável esforço), cara (desde que o reuso é factível somente se os componentes existentes possam ser facilmente adaptados para interoperabilidade) e podendo afetar a qualidade do sistema (visto que nem sempre a integração de componentes diferentes fornece bons resultados de performance e funcionalidades oferecidas) [PAY 99] [YAK 99]. Integração de Componentes COTS se refere a montagem de componentes, utilizando uma infraestrutura que fornece a ligação para formar um sistema a partir de componentes separados sem modificar o código fonte dos componentes [MEH 2000].

Por definição, qualquer projeto de desenvolvimento não pode ser totalmente COTS, pois estes não podem ser modificados pelos seus usuários por causa da ausência de código fonte e outras razões, devendo então, existir sempre algum software de integração (*glueware*, software de integração) para integrar os componentes ou adaptá-los. *Glueware* fornece a interface apropriada para um componente que está sendo integrado e serve como mediador para as interações deste componente com outros componentes. Estas atividades de integração e adaptação implicam em algum nível de entendimento dos produtos COTS (interfaces, funcionalidades, arquitetura, etc.) e envolvimento (tempo significativo de desenvolvimento de software de integração) [LOO 99] [YAK 99].

As vantagens e desvantagens de utilização de produtos COTS podem ser resumidas da seguinte maneira [LOO 99]:

Benefícios / Vantagens

- Não existe custo de desenvolvimento de componentes;
- Possivelmente existem vários fornecedores dos produtos;
- Ciclo de liberação de versões reduzido;
- O produto está mais perto do estado da arte, pois o construtor do produto provavelmente é um especialista na construção daquele tipo de produto;
- Não é necessário conhecer completamente o produto COTS internamente, isto é, sua implementação.

Custos / Desvantagens

- Perda de controle sobre o produto;
- Reposição dirigida por produtor ao invés de dirigida por consumidor, isto é, o produtor do produto define as funcionalidades e características do produto COTS;
- Consumidor possui pouca influência sobre o produto;
- Produtos restritos para domínios populares, isto é, para problemas de domínio muito específico provavelmente não existirão produtos COTS que possam ser utilizados;
- É necessário que a arquitetura do sistema seja adaptada para o produto COTS ser integrado à mesma;
- Padrões incompatíveis de segurança;

- Incompatibilidades entre produtos COTS;
- Custo da avaliação de componentes COTS;
- Custos de versões futuras e licenças dos componentes COTS.

As capacidades dos produtos COTS podem ser limitadas por dependências de uso de outros produtos de mesmo fabricante. Estas dependências são freqüentemente não obvias nas descrições das capacidades dos produtos e sua descoberta introduz atrasos e pode provocar o uso de um produto COTS em particular [LOO 99], isto é, força a utilização de produtos de mesmo fabricante.

Segundo Carney [CAR 97], existem três tipos de sistemas baseados em COTS. Esta classificação é feita de acordo com o número de produtos COTS utilizados e sua influência sobre o sistema final.

- **Sistemas de Computação Completos** (*Turnkey Systems*)- são construídos em torno de um *suíte* (conjunto) de produtos comerciais, tal como *Microsoft Office* ou *Microsoft Internet Explorer* [MIC 2001]. Somente um COTS é usado e a adaptação não faz modificações no produto COTS inicial.
- **Sistemas Intermediários** (*Intermediate Systems*) - são construídos em torno de um produto COTS (por exemplo: Oracle), mas integra outros componentes, comerciais ou desenvolvidos por completo (*in-house*).
- **Outros Sistemas** (*Other Systems*) - são construídos pela integração de vários COTS, todos no mesmo nível de importância.

5.2 Construção de ADS utilizando SGW e Outras Ferramentas Comerciais

A utilização de SGW como base para a construção de ADS já foi aplicada e proposta em alguns trabalhos [ARA 99] [BAR 2000] [BAR 2000a] [BUS 98] [CHA 97] [CHA 97a] [GRU 2000] [MAN 2001] [OCA 98].

O desenvolvimento de ADS é uma tarefa complexa, visto que ambientes deste tipo devem possuir muitas funcionalidades, e, assim sendo, devem integrar vários módulos ou ferramentas que possam suprir estas funcionalidades. As ferramentas que compõem este tipo de ambiente geralmente evoluem com o lançamento de novas versões ou mesmo quando ocorre o lançamento de novas ferramentas. Esta dinâmica de mercado para ferramentas de desenvolvimento de software torna necessário que um ADS seja capaz de integrar novas ferramentas em sua estrutura.

Uma abordagem proposta no sentido de melhorar a eficiência de construção de ambientes desta natureza é a aplicação e integração de ferramentas comercias (como Sistemas de Gerência de Workflow) de auxílio ao desenvolvimento de software para se construir um ADS [GRU 2000] [ARA 99] [BAR 2000] [BAR 2000a] [MAN 2001]. Como exemplo de ferramentas comerciais que podem ser utilizadas, cita-se: ferramentas CASE (editores diagramáticos e ferramentas de teste – como ferramentas da Rational [RAT 2001], etc.), compiladores e ambientes de programação (como o Visual Basic, Visual C++ [MIC 2001], Delphi [BOR 2002], etc.), sistemas de gerência de *workflow*

(como o *Ultimus* [ULT 2000], *Oracle Workflow Cartridge* [ORA 98], *Exchange 2000 Server* [EXC 2001], etc.), servidores de arquivos, sistemas de versionamento (como o *Microsoft SourceSafe* [MIC 2001]), ferramentas de gerência de projetos (como o Microsoft Project [MIC 2001]) e uma variedade de agentes de software [GRU 2000].

Cada ferramenta comercial integrada ao ambiente em construção fornece uma determinada funcionalidade, ou mesmo parte de uma funcionalidade. Estas funcionalidades são inerentes aos ambientes de desenvolvimento. Muitas destas ferramentas podem ser reutilizadas em outros ambientes e possivelmente em outros domínios [GRU 2000]. Assim, por exemplo, um mesmo componente pode ser utilizado como uma ferramenta de geração de documentação de software em um ADS e como uma ferramenta para produção de documentos do setor de recursos humanos de uma organização.

A utilização da abordagem descrita acima utiliza produtos comercias que já possuem amplo uso nas organizações de maneira isolada. Isto tem o propósito de aproveitar as ferramentas existentes nas organizações, que normalmente são utilizadas isoladamente, integrando-as em ambientes de desenvolvimento mais completos, que forneçam as diversas funcionalidades inerentes ao desenvolvimento de software (ou mesmo em outros domínios de aplicação). Estes ambientes iriam utilizar-se das funcionalidades individuais das ferramentas para que características como a gerência das atividades, usuários, informações, artefatos, a integração dos dados, a ativação de ferramentas, etc., sejam agregadas ao ambiente em construção.

A utilização de SGW, como um dos componentes de contrução de um ADS-CP, tem o intuito de fornecer ao ADS-CP a funcionalidade de gerência dos processos de software do ambiente. Um SGW fornece generalidade dos processos que podem ser modelados nestes sistemas, flexibilidade de adaptação a mudanças, interoperabilidade e heterogeneidade [ARA 99], isto é, um SGW fornece um paradigma de propósito geral para o suporte a execução e gerência de processos de desenvolvimento de software [CHA 97].

5.3 Alguns Trabalhos Relacionados

Manzoni [MAN 2001] propôs em seu trabalho o uso de um sistema de gerenciamento de *workflow* para apoiar o processo de desenvolvimento de sistemas. O protótipo descrito no trabalho de Manzoni foi avaliado com base em alguns requisitos considerados, por autores da área, desejáveis em um ambiente de apoio ao processo de desenvolvimento. O processo modelado neste trabalho foi baseado no processo RUP [RAT 2001], que foi expandido para alcançar os níveis 2 e 3 do modelo de maturidade CMM [SEI 2002]. O protótipo descrito por Manzoni é uma ferramenta de suporte baseada na Web, que visa auxiliar os gerentes de projeto de software nas atividades de gerenciamento e controle, e ajudar na interação e troca de informações entre os membros da equipe de desenvolvimento.

Araújo e Borges [ARA 99] apresentaram um exemplo de modelagem de um processo de desenvolvimento orientado a objetos utilizando um SGW comercial (*WebDeploy*). Neste experimento, a arquitetura baseada na Web, integrando um servidor

HTTP a um banco de dados orientado a objetos, possibilitou a modelagem, execução e monitoramento de processos de workflow (processos de software), tendo como propósito a avaliação dos SGW na modelagem e execução de processos de software, comparando suas funcionalidades com as possibilidades de suporte a este tipo de processo. Do ponto de vista de execução, os SGW, dada sua natureza cooperativa, integram seus executores na realização do processo e os auxiliam a perceber suas responsabilidades individuais e da equipe dentro do processo. Também, devido a maleabilidade dos SGW, estes podem ser utilizados para iniciar a formalização de processos nas organizações ainda imaturas e como elemento para facilitar o aprimoramento contínuo nas organizações já maduras.

Chan et. al. [CHA 97] [CHA 97a] propõem uma visão baseada em *Workflow* para o processo de desenvolvimento de software; o paradigma de *Workflow* serve como base para o suporte a execução e gerência do processo de desenvolvimento de software. A utilização de SGW possibilita a interoperabilidade, ambiente de execução distribuída, facilidade de monitoramento e gerenciamento de recursos humanos. A flexibilidade é obtida a partir do uso de um paradigma de propósito geral (SGW). Um modelo de *Workflow* enriquecido foi proposto para obter as características de interoperabilidade, execução distribuída, monitoramento e gerência de recursos humanos. Também foram definidos vários requisitos necessários para o suporte de desenvolvimento de software como processos de *workflow*; e em seus trabalhos futuros os autores propõem a modelagem e execução de vários modelos de processos de software em SGWs, como meio de teste da expressiva capacidade do modelo proposto como linguagem de especificação.

Uma abordagem de desenvolvimento para o fornecimento de ferramentas de gerência de processo que utiliza produtos de workflow como tecnologia de construção é defendida por Barnes e Gray [BAR 2000] [BAR 2000a]. Seu protótipo inicial usa um sistema de gerenciamento de banco de dados relacional (RDBMS) como um repositório de ferramentas CASE e como um simples motor de execução de processos de software. Os modelos de processo são implementados como código SQL procedural armazenado dentro do RDBMS.

Bussler [BUS 98] defende a integração de SGW e Ferramentas de Gerência de Projetos (como o Microsoft Project [MIC 2001]) para habilitar a execução controlada (planejada) de instâncias de *workflow*. Esta abordagem é necessária para controlar a iniciação não restringida de *workflow* no intuito de anular a sobrecarga de recursos humano, material e outros. A integração do SGW e Ferramentas de Gerência de Projetos consiste de duas partes: integração de esquema (os objetos do sistema são mapeados para determinação de seus relacionamentos semânticos) e integração de comportamento (baseado no mapeamento dos objetos, as mudanças de estado que ocorrem em um sistema podem ser determinadas para serem refletidas sobre o outro). A interação entre estes sistemas dá-se da seguinte maneira: antes do SGW iniciar a execução de uma instância, a ferramenta de gerência é invocada para produzir um cronograma (*schedule*); baseada neste cronograma, a ferramenta de gerência retorna para cada passo do *workflow* os recursos assinalados e a data de início daquele passo (atividade). De acordo com este padrão de interação, a execução de atividades pelo SGW ocorre de acordo com o cronograma (plano de projeto) definido pela ferramenta de gerência de projetos. Modificações no cronograma e recursos alocados são refletidos no SGW.

Grundy et. al. [GRU 2000] descreve uma abordagem baseada em componentes para o desenvolvimento de ambientes de engenharia de software. Este trabalho demonstra uma arquitetura de software baseada em componentes e um *framework* de classes Java (*JavaBeans*) que foram desenvolvidos para auxiliarem na implementação de ferramentas CASE, ambientes de projeto e Sistemas de Informação. Também demonstra uma ferramenta metaCASE que é utilizada no projeto e geração de ambientes com arquiteturas de software baseadas em componentes. As experiências relatadas neste trabalho reportam que ferramentas de software desenvolvidas a partir de arquiteturas baseadas em componentes (produtos comerciais) são geralmente mais fáceis de aprimorar e estender (integrando e empregando outras ferramentas comerciais) do que usando outras abordagens de construção.

Fuggetta [FUG 96] apresentou em um de seus trabalhos uma arquitetura típica para ADS-CP, ressaltando os principais componentes - *Interface*, *Motor de Processo* e *Repositório*.

Um aspecto importante a respeito da arquitetura de ADS-CP é que muitas vezes estes ambientes são construídos utilizando os serviços oferecidos pelos sistemas operacionais. Artefatos do processo são armazenados no sistema de arquivos e os diferentes componentes do ADS-CP são controlados e interconectados utilizando serviços de gerência de processo dos sistemas operacionais. Em outros casos, o ADS-CP é baseado em uma plataforma avançada de integração e sistemas de banco de dados [FUG 96]. A arquitetura proposta nesta seção utiliza um banco de dados como repositório e não existe, no momento, uma preocupação em como lidar com a integração de dados oriundos de diferentes ferramentas; propostas de como lidar com este problema são abordados nas seções 2.1 e 6.5.

Fuggetta [FUG 96] identificou também alguns dos principais assuntos que ainda necessitam de pesquisa na área de ADS-CP:

- Não existe um paradigma único que abranja todos os aspectos de modelagem de processo de software;
- Suporte a mudanças nos modelos de processo de software ainda é insatisfatório nos ADS-CP atuais. Neste ponto entra os SGW, que são mais genéricos e normalmente permitem a modificação de modelos de processo. Os SGW podem desempenhar o papel de motor de processo dentro de um ADS-CP construído a partir de componentes (COTS);
- Desenvolvimento de software é uma atividade criativa e não pode ser restringida por regras e procedimentos estritos. SGW permitem que os desenvolvedores de software realizem suas atividades sem restringir como estas atividades devem ser realizadas;
- Produção de software normalmente é conduzida por grupos de desenvolvedores em diferentes localidades. Os ADS-CP atuais, em sua maioria, suportam apenas um grupo de desenvolvedores trabalhando em um único local. A construção de um ADS-CP baseado na plataforma internet, isto é, construído, e acessado, como páginas *Web*, pode resolver o problema de grupos de desenvolvedores em diferentes locais (por exemplo: diferentes cidades);
- Integração de ferramentas tem sido extensivamente estudada durante os últimos anos como um assunto crítico para a construção de ADS. A

integração de ferramentas a partir de um navegador internet é abordado na seção 6.4.

5.4 Arquitetura Proposta

Fuggetta [FUG 96] apresenta uma arquitetura básica para um ADS-CP em um de seus artigos. Despeito as diferentes abordagens e paradigmas de modelagem adotados nas implementações destes ambientes, a arquitetura de um ADS-CP pode ser caracterizada pela consideração de três tipos diferentes de componentes:

1) *Interface com o Usuário*: fornece informações sobre o estado do processo e acessa as requisições dos usuários. Basicamente este componente faz o mapeamento dos eventos ocorridos no mundo real, isto é, o processo de desenvolvimento de software, para informações específicas que podem ser manipuladas pelo motor de processo. Além disso, este componente fornece ao usuário os resultados da execução do processo.
2) *Motor de Processo (Process Engine)*: executa o modelo de processo, levando em conta os eventos que ocorrem no contexto do usuário. Deve estar habilitado a ativar ferramentas e recuperar, manipular e armazenar artefatos do processo.
3) *Repositório*: dados do processo são armazenados em um repositório. Este é o banco de dados usado pelo motor de processo.

A arquitetura básica descrita na Figura 5.1 pode ser estendida e generalizada. Em alguns casos pode se: (a) utilizar múltiplos motores de processo para suportar a execução concorrente de diferentes fragmentos do mesmo modelo de processo, (b) distribuir motores de processo e interfaces com usuário em várias estações de trabalho, (c) distribuir o repositório, (d) desenvolver ambientes federados, isto é, diferentes ADS-CP compartilhando parcialmente modelos de processos e artefatos de software [FUG 96].

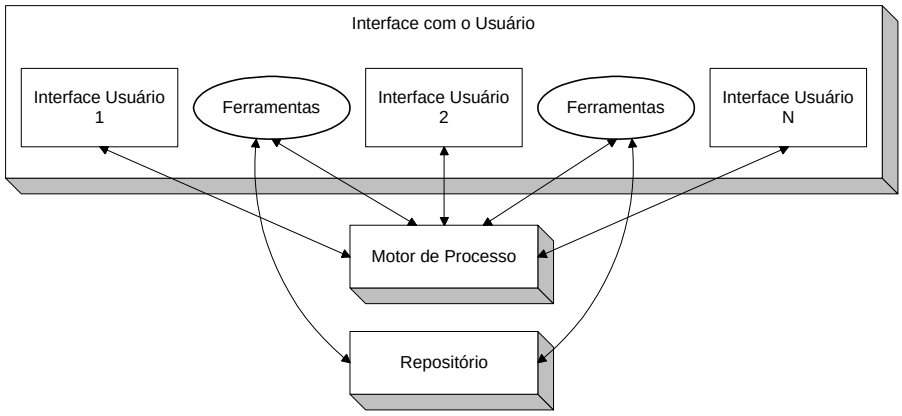

FIGURA 5.1 - Arquitetura Típica de um ADS-CP [FUG 96]

A arquitetura proposta por Fuggeta [FUG 96] servirá de base para a arquitetura de ADS (ou ADS-CP) proposta neste capítulo, fornecendo uma base conceitual para o agrupamento dos elementos que vão compor esta arquitetura. A aplicação de ferramentas comerciais (como componentes) para construção de um ADS

pode ser realizada através da integração destes produtos em um único ambiente através de alguns dispositivos como:

- A utilização da *Internet* e navegadores *Web* como plataforma de suporte do sistema, permitindo que os desenvolvedores acessem o ambiente de qualquer localização, necessitando apenas um *navegador* e acesso à *Internet*; o ambiente é apresentado e acessado como uma página da internet;
- A construção de documentos HTML e documentos que possuem mistos de HTML com códigos em linguagens de *script*, como ASP (*VBScript* + HTML) e *JavaScript*. Estes documentos são mantidos por um servidor *Web* e são responsáveis por disponibilizar as informações para os desenvolvedores e também por repassar ao ambiente as requisições destes desenvolvedores. Os documentos que contém os *scripts* são os responsáveis pela integração dos componentes COTS no ambiente, isto é, fazem o papel de **software de integração** (*glueware*), permitindo a ativação de ferramentas, *upload* de artefatos, comunicação entre o BD, o motor de processo e a interface de usuário, edição de dados de atividades e revisões, etc. Outra possível função para estes documentos é o fornecimento de serviços de *Upload*. Estes serviços permitem que o servidor *Web* armazene, em um banco de dados, arquivos originários de um computador cliente. Esta função permite que os desenvolvedores editem seus artefatos e enviem os mesmos para armazenamento em um repositório centralizado localizado no servidor *Web*;
- Os documentos ASP são os responsáveis pela interface do usuário do ambiente. Além disso, alguns destes documentos permitem que certas configurações do ambiente sejam definidas, como: os papéis desempenhados pelos desenvolvedores (podem ser adicionados e/ou alterados), as ferramentas utilizadas no processo (podem ser adicionadas novas ferramentas ao ambiente), as atividades do modelo de processo do ambiente (podem ser modificados e/ou adicionadas gabaritos, ferramentas, papéis, etc. associados as atividades do modelo de processo);
- A utilização de um sistema de banco de dados para fazer o armazenamento dos dados de controle do ambiente e dos artefatos produzidos durante o desenvolvimento de um projeto;
- A aplicação de um sistema de gerência de *workflow* (SGW) para gerenciar o processo de desenvolvimento do ambiente. O SGW é responsável pela própria modelagem, adaptação (quando necessário) e execução do modelo de processo a ser seguido. O modelo de processo define como é o assinalamento de tarefas aos desenvolvedores, envio de mensagens e fornecimento das listas de tarefas aos desenvolvedores, monitoramento da execução das atividades (início, modificações e término de atividades), definição de quais atividades devem se iniciar a partir do término de atividades anteriores, enfim, o modelo de processo define as regras e procedimentos necessários para a gerência de processos no ambiente;
- A incorporação de ferramentas utilizadas no processo de desenvolvimento de produtos de software (como compiladores, editores, ferramentas de teste, editores de texto, etc.) ao ambiente, permitindo a integração e ativação destas ferramentas a partir de interfaces do próprio ambiente. A integração de ferramentas facilita a interação do desenvolvedor com o ambiente, visto que o próprio ambiente fornece meios que facilitam a realização das atividades por fornecer a ferramenta apropriada

para execução das mesmas. A atualização de uma ferramenta ou a adição de uma nova ferramenta ao ambiente também é possível;

- A disponibilização dos artefatos gerados em um projeto através do servidor *Web*. Os artefatos seriam disponibilizados através de um hiperdocumento (gerado dinamicamente a partir dos artefatos contidos no repositório do ambiente) contendo todos os artefatos produzidos em um projeto de software, permitindo assim o acesso dos desenvolvedores aos artefatos já concluídos;

Na Figura 5.2 a arquitetura proposta pode ser visualizada graficamente. Esta arquitetura demonstra os componentes ao nível de meios de comunicação entre os mesmos e o acesso dos desenvolvedores ao ambiente, e está baseada na arquitetura proposta por Fuggetta [FUG 96], agrupando os elementos de acordo com esta última (Figura 5.1).

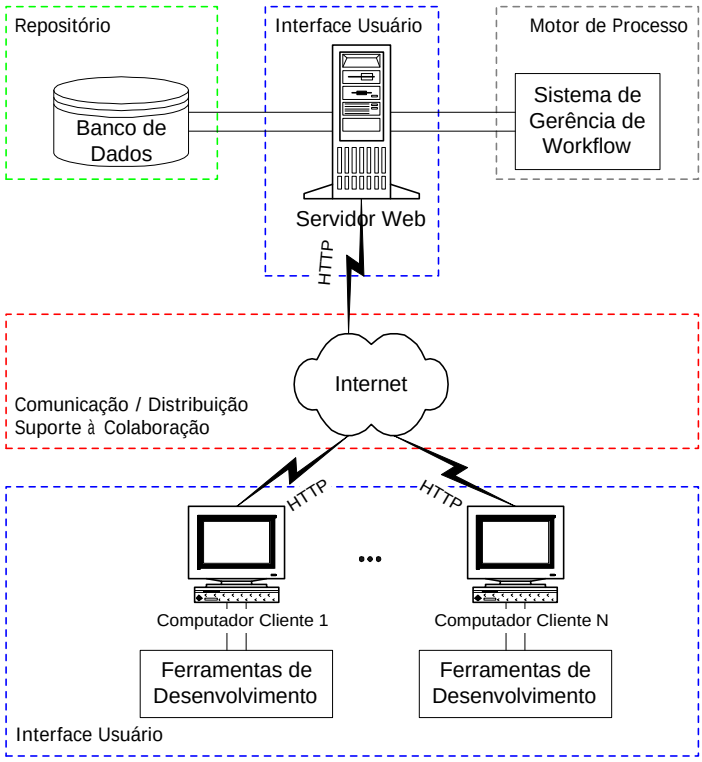

FIGURA 5.2 - Arquitetura Proposta para ADS

Basicamente, a arquitetura é composta pelos seguintes elementos:

- **Servidor *Web***: esta é a máquina onde é instalado fisicamente o ambiente. O Servidor *Web* mantém o banco de dados, o sistema de gerência de *Workflow* e os documentos associados ao ambiente (HTML mais *scripts*). Analisando este elemento em relação à arquitetura típica descrita acima (Figura 5.1), o Servidor *Web* faz parte do componente *Interface com o Usuário*, visto que este fornece os documentos e dados necessários para a interação dos desenvolvedores com o ambiente, que é realizada através de computadores clientes.
- **Banco de Dados**: O banco de dados (BD) fica localizado no servidor *Web*. Sua função é armazenar os dados referentes ao processo (dados de controle), o modelo de processo (esquema de processo que é instanciado ao início de cada novo projeto e

executado pelo SGW) e os artefatos produzidos durante um projeto. O BD também se encarrega de manter os dados em um estado consistente. O BD é equivalente ao *Repositório* descrito na Figura 5.1.

- **Sistema de Gerência de *Workflow***: localizado também no servidor *Web*, o SGW fica encarregado de controlar a execução, assinalamento, monitoramento, etc. das atividades realizadas dentro do ambiente. O SGW também interage com o BD, visto que é no BD onde o modelo de processo e os dados de controle são armazenados. O SGW, na realidade, desempenha o mesmo papel do *Motor de Processo* da arquitetura proposta por Fuggetta (Figura 5.1).
- **Computadores Cliente**: através de computadores que possuem navegadores e acesso a *Internet* é que os desenvolvedores podem interagir com o ambiente. Por meio dos navegadores os desenvolvedores enviam requisições ao servidor *Web*, que por sua vez, utilizando-se dos documentos de interface do ambiente enviam aos navegadores clientes as páginas de acesso e recuperação dos dados dos projetos. Estes elementos (computadores cliente), se comparados à arquitetura de Fuggetta [FUG 96], fazem parte do componente *Interface com o Usuário* (Figura 5.1). Observa-se que as ferramentas de desenvolvimento utilizadas durante o processo estão instaladas nos computadores cliente; assim, cada usuário do ambiente pode utilizar a ferramenta que desejar, atentando somente ao fato de que os artefatos resultantes a partir do uso destas ferramentas devem ser carregados para o *Servidor Web* em um formato de documento que possa ser utilizado por outros desenvolvedores em computadores diferentes (computadores cliente). Neste ponto se ressalta a importância da integração de dados e como ferramentas podem utilizar dados oriundos de outras ferramentas; estes tópicos são abordados nas seções 2.1 e 6.5.
- ***Internet***: a *Internet* age como uma plataforma de comunicação entre os computadores clientes (navegadores clientes) e o servidor *Web*. A *Internet* permite a colaboração entre os desenvolvedores, permitindo que os desenvolvedores visualizem as suas atividades, atualizem os dados do projeto, atualizem ou criem os artefatos associados a um projeto, etc. O assinalamento de atividades para os desenvolvedores é realizado pelo SGW, mas o acesso às listas de tarefas é feito através dos navegadores clientes, que utilizam a *Internet* para obter os dados do Servidor *Web* (que contém o BD e o SGW). Nos dias de hoje os processos de software (e processos em geral) tendem a ser cada vez mais distribuídos. Devido às novas tecnologias de suporte (Java, CORBA, Internet), o trabalho deveria ser geograficamente espalhado e os agentes não necessitam estar no mesmo prédio ou mesmo na mesma cidade [OCA 98]. Atualmente os SGW utilizam a *Internet* como uma maneira de distribuir o trabalho; da mesma maneira alguns protótipos de ADS-CPs usam também a *Internet* para suportar os processos [OCA 98]. A *Internet*, como um componente de comunicação, distribuição e colaboração, não está prevista na arquitetura descrita por Fuggetta [FUG96]. Desta forma, o uso da *Internet* torna o ambiente não atrelado, fisicamente falando, ao setor de desenvolvimento das empresas, permitindo que os desenvolvedores trabalhem em qualquer lugar do planeta.
- **Ferramentas de Desenvolvimento**: as ferramentas utilizadas pelos desenvolvedores são ativadas pelo próprio navegador *Web*. Estas ferramentas devem estar cadastradas no ambiente e as informações destes cadastros são armazenadas no repositório do

ambiente. Para cada computador cliente que acessa o ambiente, existe um conjunto de ferramentas correspondente aquele computador (identificado pelo IP desta máquina). Outras maneiras de identificação devem ser definidas futuramente.

No intuito de descrever melhor a arquitetura proposta, o diagrama de componentes [FOW 2000] [LAR 2000] [BOO 2000] da arquitetura é apresentado na Figura 5.3. Esta figura mostra, em maior detalhe, os componentes participantes da arquitetura e os seus relacionamentos e dependências.

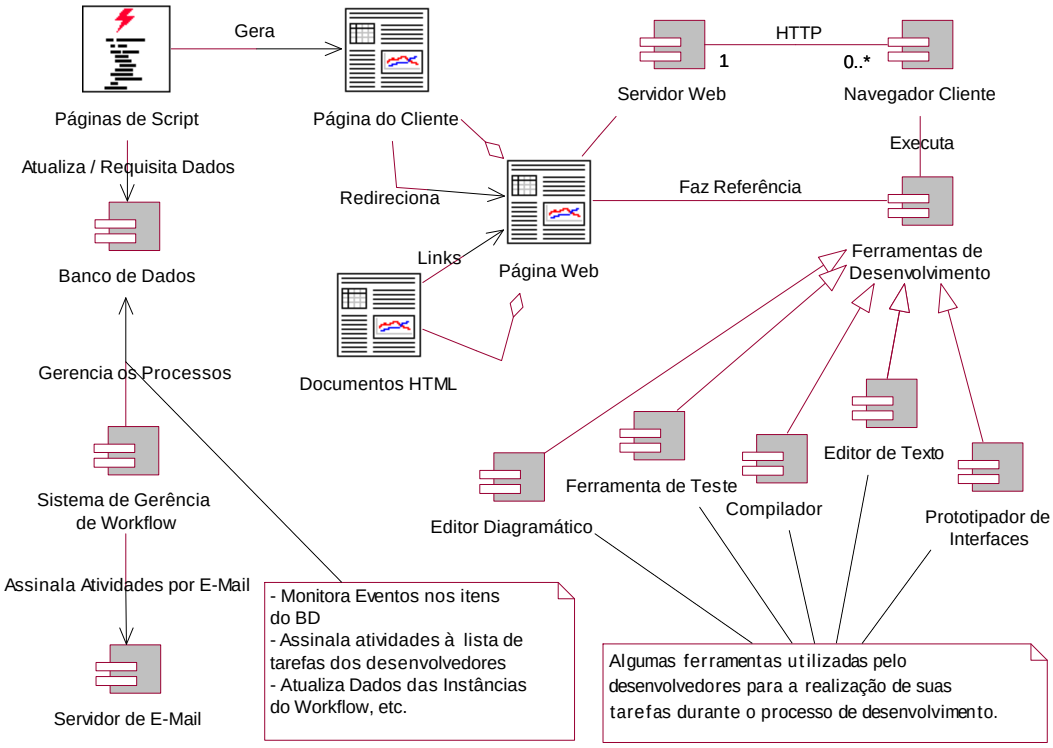

FIGURA 5.3 - Modelo de Componentes para ADS

As *páginas de script* (documentos ASP) interagem com o *banco de dados* atualizando e requisitando dados do ambiente para geração dinâmica de páginas (no formato HTML).

O *Sistema de Gerência de Workflow* controla as instâncias do modelo de processo em execução em determinado momento. Cada projeto associado ao ambiente possui uma instância deste modelo de processo. O SGW atualiza os dados de controle no banco de dados a partir de qualquer modificação ocorrida. Quando uma nova atividade é assinalada a um determinado desenvolvedor, uma mensagem eletrônica é enviada (através do *servidor de E-Mail*) informando este desenvolvedor.

Os navegadores *clientes* se comunicam com o *servidor Web* através do protocolo HTTP (*Internet*). A *página Web* que é enviada ao *navegador cliente* é gerada dinamicamente através das *páginas de script* ou é constituída por *documentos HTML* localizados no *servidor Web*. Estes *documentos HTML* podem conter informações gerais sobre o processo, atividades, gabaritos, etc.

A *página Web* é o documento que é visualizado no *navegador cliente*. Este documento faz referências para as *ferramentas de desenvolvimento* utilizadas pelos desenvolvedores. Uma *página Web* geralmente é composta pela agregação de *documentos HTML* contidos no servidor e por páginas construídas dinamicamente através das *páginas de script*.

O *navegador cliente* faz a interface entre os desenvolvedores e o ambiente. Este também é responsável pela execução (ativação) das ferramentas que serão utilizadas pelos desenvolvedores.

Na Figura 5.4 é mostrado um diagrama de seqüência que descreve um exemplo de relacionamentos de controle entre os componentes da arquitetura, de ativação de ferramentas de desenvolvimento, de troca de informações entre os componentes (principalmente em relação ao BD), de inicialização de atividade a partir do modelo de processo do SGW (gerência do projeto).

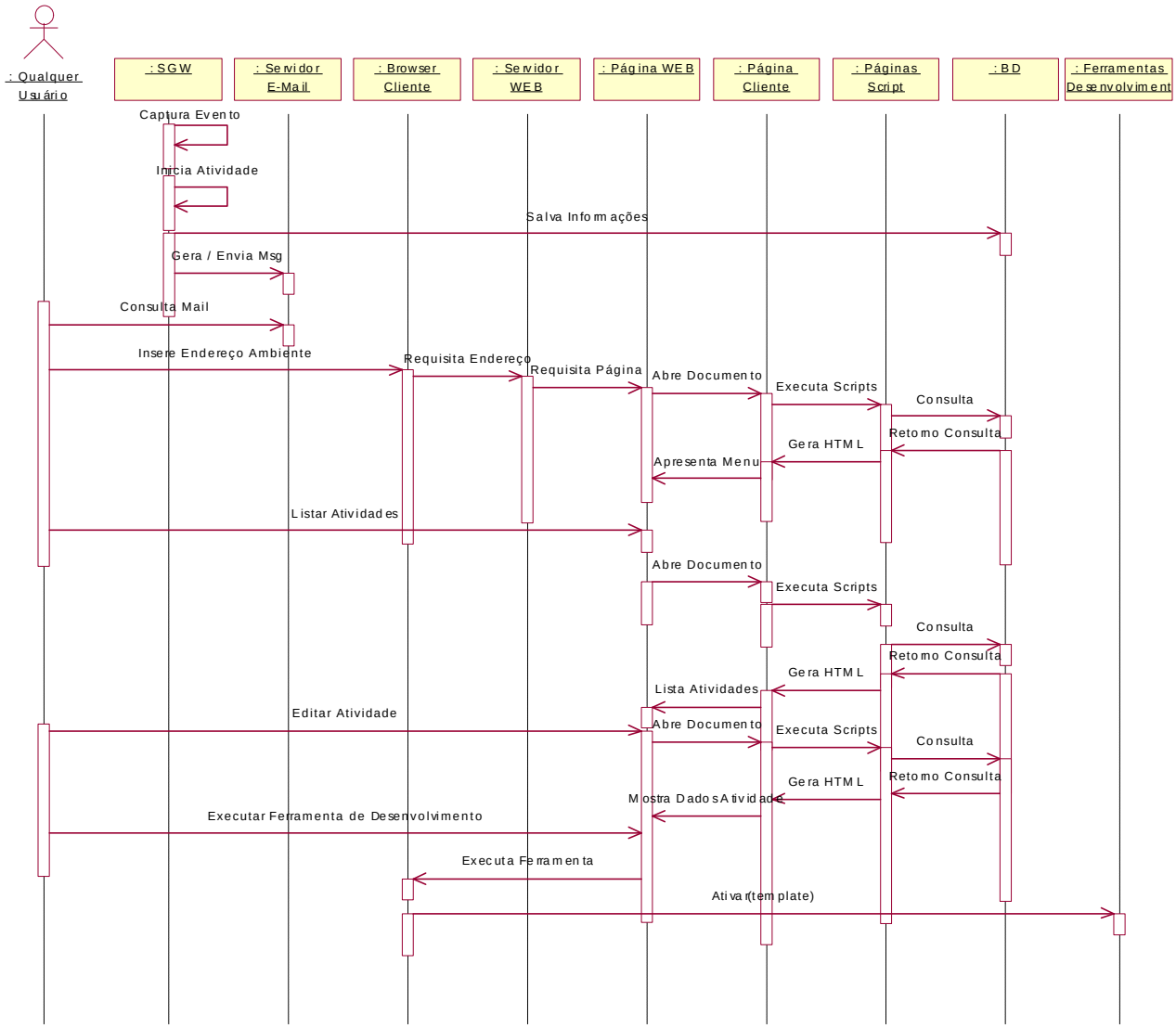

FIGURA 5.4 – Diagrama de Seqüência mostrando alguns relacionamentos dinâmicos dos Componentes da Arquitetura

O exemplo da Figura 5.4 mostra a seguinte seqüência de eventos: a captura de um evento de término de atividade pelo SGW e início de nova atividade, inclusive com o armazenamento de informações relevantes no BD; um usuário do ambiente recebe uma mensagem de correio eletrônico e em seguida faz a verificação de sua lista de atividades no ambiente; o usuário abre a interface de edição de atividade e em seguida clica no *link* de ferramenta para ativar a ferramenta associada a esta atividade.

5.5 Vantagens e Desvantagens de Arquiteturas Baseadas na Aplicação de Ferramentas Comerciais como Unidades de Construção

Ferramentas de Engenharia de Software são usualmente aplicações complexas. Muitas vezes necessitando suporte a múltiplas visões com técnicas de gerenciamento de consistência apropriadas, suporte a múltiplos usuários e suporte a graus apropriados de integração com outras ferramentas, geralmente de terceiros [GRU 2000].

A utilização de produtos comerciais (COTS) está se tornando muito popular [OBE 98]. A utilização de arquiteturas baseadas em ferramentas ou produtos comerciais para a construção de ambientes de engenharia de software tornam os mesmos mais fáceis de evoluir e estender, permitindo a integração com outras ferramentas e o emprego de ferramentas construídas a partir de outras abordagens [GRU 2000].

Um dos objetivos associados ao crescente uso de produtos comerciais como componentes para construção de ambientes integrados, é a possibilidade de se utilizar, a um preço razoável e com uma tecnologia avançada, algo (componente) que já realize as funções necessárias para um determinado ambiente [OBE 98]. O custo associado à construção de ADS´s (que são ambientes especializados) é geralmente alto, de modo que usar ferramentas prontas, existentes no mercado, que forneçam as funcionalidades necessárias é mais barato, e reduz o tempo de desenvolvimento [BAR 2000a]. Os objetivos da aplicação de produtos comerciais como componentes são [OBE 98] [CAR 97]:

- Melhorar a qualidade e performance dos sistemas. Entretanto, estas características vão depender muito do software de integração utilizado para “agrupar” os componentes.
- Tornar o desenvolvimento de sistemas mais rápido, por utilizar produtos já existentes que forneçam funcionalidades iguais, ou pelo menos parecidas, com as que são necessárias para um software ou ambiente em construção.
- Construção de ambientes mais baratos. Este objetivo depende muito dos componentes escolhidos e do software de integração utilizado. Se forem utilizados componentes de difícil interfaceamento, o tempo de desenvolvimento de software de integração será demasiadamente grande e, possivelmente, oferecendo problemas de performance e compatibilidade.

Construir um ADS a partir do "zero" é uma tarefa complexa, geralmente envolvendo muitos profissionais, longos cronogramas e custos altos. Se forem utilizados ferramentas comerciais para construção (integração das ferramentas em um único ambiente) de ADSs, a construção destes ambientes pode ser mais rápida (visto que integrar ferramentas prontas geralmente é mais fácil do que construir ferramentas por inteiro), envolvendo um menor número de profissionais (proporcionando um menor custo de recursos humanos) e fornecendo, a princípio, produtos mais confiáveis, pois produtos já existentes no mercado geralmente possuem menos "*bugs*", dada a experimentação realizada sobre estes produtos e a liberação de novas versões com correção de erros.

Ferramentas (ou mesmo ambientes) de software baseadas em componentes facilitam mais a integração, se comparada com outras arquiteturas para construção de ferramentas, já que os componentes geralmente possuem interfaces bem definidas com mecanismos de monitoramento de eventos embutido. Estes ambientes baseados na integração de ferramentas comerciais tendem a serem construídos com características que propiciam reuso e extensão [GRU 2000].

A Tabela 5.1 faz um comparativo entre a abordagem de construção de ADS baseada na integração de ferramentas comerciais (COTS) e a abordagem de construção por completo (do “zero”; *from scratch*), ressaltando algumas características importantes associadas a construção de ADS, e mesmo ambientes para outros domínios de aplicação.

TABELA 5.1 - Comparativo entre a abordagem baseada na integração de ferramentas comerciais e construção por completo (Continua)

Característica	Integração de Ferramentas Comerciais	Construção por Completo
- Qualidade e Performance	Produtos COTS são bem testados [PAY 99] [YAK 99], fornecendo produtos mais confiáveis. Ao longo do tempo de uso, produtos COTS vão melhorando [MOR 2000]. Software de Integração pode ser um problema de custo e tempo de desenvolvimento se os produtos COTS não oferecerem interfaces bem definidas [LOO 99] [PAY 99] [YAK 99]. Por terem escopo de atuação bem definido, e normalmente restritos, produtos COTS normalmente oferecem melhor qualidade e performance.	Ambientes construídos desde o projeto (*from scratch*) também são construídos com propósitos e escopo de atuação bem definido, mas trata-se de sistemas completos, sendo assim, mais difícil se alcançar estes objetivos com qualidade e performance em um prazo e custo competitivos.
- Tempo de Construção e Cronograma	Ferramentas comerciais são implementações de funcionalidades bem definidas, geralmente a integração destas demanda um menor tempo de construção. Estes componentes não precisam ser contruídos, reduzindo custos [LOO 99] [PAY 99]. Desenvolvimento de *glueware* pode demandar considerável tempo de construção e até mesmo ocasionar atrasos de cronograma [PAY 99] [YAK 99]. Componentes mal documentados e dependentes de fabricante pode também causar atrasos [LOO 99]. Será necessário se gastar algum tempo de desenvolvimento para a seleção dos produtos COTS [LOO 99]	As funcionalidades oferecidas por um produto COTS em uma abordagem de construção baseada em COTS devem ser implementadas em sua totalidade em uma abordagem de construção por completo, tornando a construção de um ambiente muito demorada, dada a complexidade e tamanho de ambientes como ADS e outros.
- Manutenção	Devido a dinâmica de mercado, novos produtos são lançados em curtos períodos de tempo. As vezes, a economia feita com o uso de um produto COTS pode ser perdida na manutenção do produto para mantê-lo atualizado [OBE 98]. Manutenção corretiva a partir do encontro de erros ocorre geralmente sobre o software de integração [LOO 99] [PAY 99] [YAK 99], dificilmente ocorrendo manutenção no próprio produto COTS, a não ser, como já mencionado, em casos de atualização de produtos COTS.	Qualquer atualização e/ou melhoramento, ou mesmo a correção de erros, provoca a modificação de código fonte, necessitando tempo de desenvolvedores para implementar estas atualizações / modificações.

TABELA 5.1 - Comparativo entre a abordagem baseada na integração de ferramentas comerciais e construção por completo (Continuação)

- Custo de Construção	A utilização de produtos COTS tendem a reduzir o orçamento dos projetos [MOR 2000] e tempo de desenvolvimento dos ambientes (reduzindo também o custo com recursos humanos). Produtos COTS já existentes nas organizações podem ser integrados a um ambiente e / ou reutilizados em outros ambientes [LOO 99] [GRU 2000]. Custos de avaliação e seleção de produtos COTS e aquisição de licenças e versões futuras de COTS são acrescentados ao orçamento [LOO 99].	Custo mais alto, tarefa muito dispendiosa em termos de mão de obra e cronogramas muito longos quando se está construindo sistemas complexos. Dependendo do sistema que se pretende construir, às vezes, por questões de custo, é melhor desenvolver o software a partir do nada.
- Extensibilidade	Ambientes desenvolvidos a partir de COTS tendem a ser mais facilmente melhorado e estendidos do que utilizando outras abordagens [GRU 2000].	E necessário que uma nova versão do software seja liberada, necessitando para isso nova implementação para extensão.
- Tamanho e Complexidade	O tamanho e complexidade que os atuais sistemas (incluindo ADS) demandam tornam difícil e cara a construção de sistemas a partir do "zero", portanto, o emprego de produtos COTS tem sido uma solução normalmente acatada [CHU 2002]. Pode-se perder o controle sobre os produtos COTS devido a problemas de comunicação entre o produto COTS e o software de integração, ou mesmo entre produtos COTS diferentes [LOO 99].	Quanto maior a complexidade de um sistema, mais demorado, caro e arriscado é um esforço de desenvolvimento de software.
- Estado da Arte	Um produto COTS, normalmente fabricado por um especialista naquele tipo de software, fornece produtos e funcionalidades mais avançadas, mais perto do "estado da arte" e de produtos em pesquisa [LOO 99].	Um produto pode ser construído com objetivo de se manter o mais próximo possível do estado da arte, mas como o tempo de desenvolvimento associado a este normalmente é longo, e possivelmente os desenvolvedores não possuem experiência em determinado domínio de aplicação, o produto, quando pronto, acaba por ficando obsoleto comparado aos produtos oferecidos por fabricantes especialistas.
- Domínio de Aplicação	Produtos COTS possuem vários fornecedores com os mais variados tipos de produtos, abrangendo um gama grande de domínios de aplicação é não é necessário se conhecer o produto internamente [LOO 99]. As funcionalidades dos COTS são definidas pelos fabricantes, e não de acordo com a necessidade dos usuários daquele produto [LOO 99].	O produto é construído com base em requisitos específicos de um domínio. O produto final, quando produzido com a aplicação de técnicas de Engenharia de Software, fornecem as funcionalidades específicas necessárias, algumas vezes a um custo menor. As funcionalidades são definidas pelos usuários (consumidores dos produtos).

Uma das desvantagens mais sérias da aplicação de ferramentas comerciais de diferentes fabricantes para a construção de ADS, ao invés de desenvolvimento específico, é a possível perda de controle sobre o produto final e a falta de associação com outros produtos, visto que o fabricante não libera o código fonte dos produtos e, algumas vezes, os produtos não estão de acordo com os padrões de interface; estes problemas levam a dificuldades, senão a impossibilidade, de construção de software de integração. Produtos comerciais geralmente são atualizados (liberação de novas versões) periodicamente, em espaços de tempo cada vez mais curtos, assim, utilizar estes produtos pode levar a necessidade de se atualizar as versões destes produtos periodicamente, no intuito de se manter os produtos atualizados [OBE 98], causando então dispesas extras aos usuários.

Os principais benefícios da utilização de produtos COTS no desenvolvimento de ambientes complexos como ADS-CP são resumidos a seguir:

- A redução de tempo de desenvolvimento: desenvolver software rapidamente pode significar o ganho de uma grande fatia de mercado caso produtos similares ainda estão em desenvolvimento. A avaliação e seleção de produtos COTS deve ser cuidadosamente realizada, optanto sempre por produtos de interfaces bem definidas

e, se possível, não atreladas a um fabricante específico [LOO 99]; e o processo de desenvolvimento utilizado para ambientes baseados em COTS deve levar em consideração atividades como seleção de produtos COTS, implementação de software de integração, etc. [YAK 99] [PAY 99] [MOR 2000].

- A redução dos custos de desenvolvimento: um dos objetivos de utilização de ferramentas comerciais é a redução do tempo de desenvolvimento. Esta redução proporciona uma economia nos gastos de recursos humanos de um projeto. Produtos COTS normalmente proporcionam a redução dos custos de orçamento [MOR 2000].
- A utilização de produtos já existentes nas organizações: a integração de ferramentas de funcionalidades específicas em um ambientes complexos como ADS pode ser realizada através da abordagem proposta neste capítulo. Ferramentas já utilizadas pelas organizações podem ser reutilizadas na construção de outros ambientes (ADS), diminuindo custos com licenciamento de produtos [GRU 2000] [LOO 99].
- Utilização de funcionalidades de produtos COTS: a implementação de funcionalidades como gerência e assinalamento de atividades são difíceis de implementar em ADS-CP se construídas a partir do "zero". As funcionalidades fornecidas por um SGW podem suprir esta necessidade. Um SGW pode ser integrado, através de software de integração (que no caso da arquitetura proposta neste capítulo é realizado através de código ASP), e utilizado como o motor de processo de um ADS-CP [FUG 96] [CHU 2002]. Na realidade, qualquer funcionalidade necessária a um ADS-CP, ou outro tipo de ambiente, pode ser sanada por um produto COTS, desde que este forneça funcionalidades iguais ou ao menos semelhantes.

5.6 Comparação com os Trabalhos Relacionados

Nesta seção uma comparação entre os trabalhos relacionados descritos na seção 5.3 e a arquitetura proposta na seção 5.4 será realizada.

O trabalho de Manzoni [MAN 2001] está diretamente ligado a este trabalho (visto que Manzoni, assim como o autor deste trabalho, também faz parte do grupo de pesquisa AMADEUS [AMA 2002]) e pode ser visto como base para a realização do mesmo. A arquitetura do protótipo implementado por Manzoni utiliza o SGW *Microsoft Exchange 2000 Server* e todos os mecanismos associados a este produto (como o *Active Directory* do *Windows 2000 Server*, o *Web Storage System* e páginas ASP como mecanismo de acesso e atualização dos dados). A arquitetura do protótipo de Manzoni é similar à descrita na Figura 5.2. O protótipo de Manzoni não permite a adição de novos papeis, ou funções, que podem ser desempenhados pelos desenvolvedores associados aos ambientes, não permite que as atividades do processo sejam configuradas em relação aos papeis, gabaritos e manuais de orientação associados, não ativa ferramentas de terceiros (outros fabricantes senão *Microsoft*), não faz a integração dos artefatos

produzidos e não permite o *upload*[†] dos artefatos localizados nas máquinas utilizadas pelos desenvolvedores para o servidor *Web* onde está instalado o protótipo.

Araújo et. al. [ARA 99] utiliza em seu experimento um SGW chamado *WebDeploy*, juntamente com um servidor HTTP e um banco de dados orientado a objetos. Comparando com a arquitetura da Figura 5.2 temos várias semelhanças, como: a utilização de um SGW (como um "gerente dos processos"), um servidor HTTP (servidor *Web*) e um banco de dados. No entanto, não é demosntrado como a integração de ferramentas e de dados pode ser implementada. Também não é relatada a possibilidade de adição e alteração de papéis e configuração de atividades.

Chan et. al. [CHA 97] [CHA 97a] propõem uma visão baseada em *Workflow* para o processo de desenvolvimento de software. Também foi proposto um modelo enriquecido de *workflow* para permitir esta visão e foram descritas algumas características associadas a ambientes de desenvolvimento de software. Nos trabalhos de Chan e Leung não é mostrado como integrar SGW, BDs e ferramentas de desenvolvimento de software em um ambiente como ADS.

Barnes e Gray [BAR 2000] [BAR 2000a] utilizam em seus trabalhos um sistema de gerencia de banco de dados como um repositório CASE. Este sistema também faz o papel de um simples (pseudo) SGW. O processo de software é armazenando como código procedural escrito em declarações SQL que "imitam" o comportamento de um SGW. Assim, o sistema de banco de dados faz o papel, simultaneamente, de um repositório de dados e de um motor de processo. Esta abordagem restringe a capacidade de modelagem de processos de software.

Bussler [BUS 98] defende a integração de SGW e ferramentas de gerência de projetos para habilitar a execução controlada de instâncias de *workflow*. A arquitetura proposta na Figura 5.2 não utiliza nenhuma ferramenta de gerência de projetos, embora fosse muito interessante acrescentar uma ferramenta deste tipo para a realização do planejamento das atividades, relatórios sobre atividades atrasadas e o impacto deste atraso sobre o projeto como um todo. A utilização de uma ferramenta de gerência na arquitetura da Figura 5.2 possibilitaria que o gerente de processo gerasse o cronograma de atividades nesta ferramenta (no protótipo atual o gerente de projeto necessita gerar este cronograma manualmente) e este seria utilizado pelo SGW para o assinalamento de atividades. O término de atividade capturado pelo SGW seria reportado a ferramenta de gerência, que por sua vez poderia gerar novo cronograma levando em conta possíveis atrasos.

Grundy et. al. [GRU 2000] descreve a aplicação de componentes para a construção de ADSs. Não é proposta nenhuma arquitetura generalizada; são demonstradas as experiências adquiridas na aplicação de diferentes componentes para a construção de um protótipo de ADS. Outro aspecto é que os componentes utilizados foram desenvolvidos anteriormente pelo mesmo grupo de pesquisa ou grupos associados. Desta maneira, não foram aplicados produtos prontos (produtos COTS) disponíveis comercialmente, diminuindo a chance de ocorrência de problemas de integração de produtos.

Fuggetta [FUG 96] apresentou uma arquitetura típica para ADS-CP. Esta arquitetura foi estendida e modificada pela utilização de produtos COTS como

[†] *Upload* é uma operação que permite que um arquivo seja copiado de um computador cliente para um servidor. É a operação inversa do *Download*.

componentes da arquitetura, gerando a arquitetura proposta neste capítulo. A plataforma *internet* não é levada em consideração na arquitetura de Fuggetta; inclusive, a arquitetura de Fuggetta, não aborda aspectos de comunicação e distribuição de trabalho em grupos geograficamente distantes de desenvolvedores.

5.7 Adequação da Arquitetura Proposta aos Requisitos definidos na Seção 2.4

Nesta seção a arquitetura proposta durante este capítulo é avaliada em relação a aderência dos requisitos necessários para ADS descritos na seção 2.4.

- **Suporte ao Gerenciamento** - Os dados referentes aos processos e que são utilizados e atualizados pelo SGW podem ser recuperados por meio das páginas ASP, que também podem requisitar e atualizar estes dados. Estes dados são mantidos no repositório (banco de dados) do ambiente. As páginas ASP acessam o repositório através de consultas SQL. A utilização das interfaces *Internet* (*navegador*) e documentos ASP (*VBScript* e *JavaScript*) possibilitam a recuperação dos dados referentes a um determinado projeto, permitindo, desta maneira, que qualquer modificação ocorrida no processo - como assinalamento de atividades, disseminação de informações necessárias à execução das atividades e coleta de informações geradas como resultado da execução da atividade pelo usuário - seja dinamicamente atualizada “aos olhos” do desenvolvedor. Os dados referentes ao processo e que são utilizados pelo SGW podem ser recuperados também por meio de páginas ASP.
- **Suporte aos Eventos do Processo –** O SGW integrado ao ambiente faz o monitoramento dos eventos. Estes eventos geram o disparo de ações correspondentes, como: assinalar atividade a um trabalhador (disponibilizando também as informações, gabaritos, manuais de orientação, etc. referentes a esta atividade) após o término de uma atividade anterior, enviar *e-mail* notificando o gerente sobre a necessidade de sua intervenção (caso não exista a definição de desenvolvedor responsável por atividade que deve ser iniciada imediatamente), etc.
- **Flexibilidade** – O SGW permite que o processo seja modificado quando necessário. Isto é feito através das linguagens de modelagem de processos (PML) - geralmente disponibilizadas como ferramentas gráficas que utilizam formalismos (como diagramas de estados e de atividades [FOW 2000], redes de Petri [BAN 96], etc.) para definição dos processos - que normalmente acompanham os SGW. **Processos** definem as etapas, seqüências e dependências entre as atividades que devem ser realizadas durante um projeto. Esta característica de modificação dos processos permite a adaptabilidade dos mesmos de acordo com os projetos em desenvolvimento ou com os processos adaptados definidos pelas organizações. É possível também que a metodologia de desenvolvimento seja modificada, pela adequação dos artefatos associados, dos manuais de orientação, das ferramentas associadas às atividades, etc. Através de utilização de documentos estilo ASP que possibilitam que o ambiente seja configurado em termos de usuários, papéis desempenhados, ferramentas utilizadas e atividades a serem desempenhadas. **Metodologia** é a aplicação de métodos e técnicas para a construção de modelos, a partir de utilização de determinada notação que produzirá uma descrição de produtos de software. A metodologia utilizada em um ADS depende das ferramentas

integradas a este ambiente. Dependendo das ferramentas do ambiente teremos técnicas e notações diferentes. Como o ambiente se propõe a permitir a integração de ferramentas de diferentes fabricantes, basta se utilizar ferramentas que apliquem as técnicas e notações apropriadas. Um problema a se resolver é a integração entre as ferramentas, isto é, como as ferramentas podem trocar informações, como abrir artefatos (documentos, diagramas, código fonte, etc.) gerados em uma ferramenta em outra, etc. Uma maneira é a utilização de formatos padrão de documentos, por exemplo, documentos XML [XML 2001].

- **Extensibilidade –** O ambiente possibilita a inserção de novas ferramentas de apoio ao desenvolvimento de software através do cadastro de novas ferramentas (e suas respectivas informações) através das interfaces de acesso (páginas ASP), que por sua vez armazenam estas informações no repositório do ADS.
- **Reusabilidade –** O ambiente mantém um conjunto de documentos e gabaritos que podem ser reutilizados em qualquer projeto. O processo de software definido no SGW é executado em separado para cada novo projeto que se inicia. O ambiente também permite que os artefatos produzidos em projetos anteriores sejam acessados via navegador, através da construção, em tempo de execução, de um hiperdocumento contendo todos os artefatos referentes a um projeto, permitindo que estes artefatos sejam acessados, alterados reutilizados em outro projeto semelhante, diminuindo o tempo e os gastos de construção do novo produto e também adiantando os cronogramas.
- **Colaboração -** Um dos requisitos mais importantes de um ADS é a colaboração. O desenvolvimento de um novo produto de software necessita ser realizado por grupos de profissionais trabalhando em colaboração. A *Web* fornece uma maneira de se cooperar no desenvolvimento de software. Utilizando-se de um navegador, acesso à *Internet* e a URL de um servidor *Web* com um ADS instalado, vários desenvolvedores podem estar atuando em diversas localidades ao redor do planeta em um mesmo projeto. A arquitetura básica de uma aplicação *Web* inclui navegadores, uma rede e um servidor Web. Os navegadores fazem requisições de páginas *Web* de um servidor. Algumas páginas incluem *scripts* do lado cliente que são interpretados pelo navegador e que podem acessar dados de arquivos, bancos de dados, ou mesmo interagir com SGW [COJ 99]. A arquitetura do ambiente proposto possui um servidor *Web*, uma rede de comunicação (*Internet*), navegadores *Web* (clientes), um sistema de banco de dados e documentos de páginas *Web*. Um artefato de software, resultado de alguma atividade, pode ser salvo no servidor pelo seu desenvolvedor. Se outro desenvolvedor necessitar deste artefato para realizar outra atividade, este pode acessar o servidor e acessá-lo através da visualização dos vários artefatos já produzidos em um projeto (em forma de um hiperdocumento).
- **Fácil de Usar -** A utilização da *Internet* já é uma realidade para os profissionais de desenvolvimento de software, assim, utilizar-se de interfaces padrões da *Web* (páginas que possuem *links* para outras páginas), como as encontradas nos navegadores, para se acessar um ADS é uma facilidade desejada. A arquitetura do ambiente utiliza, além da geração de listas de atividades por desenvolvedor, também *e-mail* eletrônico como meio de comunicação entre o ambiente e os desenvolvedores.
- **Monitoramento –** Os dados referentes a um projeto são mantidos no repositório do ADS. Estes dados podem ser analisados no sentido de se coletar métricas. A coleção

de métricas não está contemplada na arquitetura proposta. Isto poderia ser conseguido através da integração de uma ferramenta de coleta / análise de métricas na arquitetura proposta neste capítulo.

- **Consistência –** todos dados do ambientes são armazenados em um banco de dados, que atua como o repositório do ADS. Este banco de dados fica responsável pelo controle das transações de criação, atualização e recuperação de dados do ambiente.
- **Verificação -** Ferramentas de modelagem de processos, as quais estão normalmente incorporadas aos SGW, permitem que processos sejam definidos, em sua maioria, visualmente, possibilitando a verificação dos processos através de dispositivos de depuração e simulação. O *Exchange 2000 Server* (E2K), SGW utilizado para construção do protótipo deste trabalho, não permite a depuração/simulação de processos.
- **Integração de Controle –** o repositório mantém as informações referentes às ferramentas do ambiente. Estas informações são utilizadas para a criação dos *links* de ativação destas ferramentas. Quando uma atividade não possui nenhuma ferramenta associada, o ambiente pode fornecer, através de um *link*, uma ferramenta padrão para a execução da mesma.
- **Integração de Dados -** O servidor *Web* possui todos os documentos *Web* que cuidam da parte gerencial do ambiente. Estes documentos acessam um sistema de banco de dados que, por sua vez, armazena dados de controle do processo e do ambiente e os artefatos produzidos durante o processo. O ambiente possibilita a integração dos artefatos produzidos durante um projeto através de dispositivos de *Upload* e armazenamento no servidor.

5.8 Modelagem e Execução de Processos de Software

A modelagem de um modelo de processo é realizada utilizando as linguagens de definição de processos existentes nos SGW. Um processo de software é definido e o mesmo é modelado na notação do SGW utilizado.

O SGW faz o monitoramento dos eventos e dispara as ações associadas (como o assinalamento e início de nova atividade) quando condições prévias são alcançadas. Estas condições e ações são todas definidas na notação da linguagem de definição de processos de workflow do SGW.

Quando uma nova atividade é assinalada, o SGW instância esta atividade e, de acordo com a definição da atividade, o próprio ambiente fornece ao desenvolvedor as informações necessárias sobre a atividade, o gabarito associado à atividade (servindo de modelo e guia para a execução da atividade) e ferramenta que pode ser utilizada para a realização desta atividade.

Todas estas informações referentes às atividades como gabaritos, ferramentas e documentos associados - além de informações sobre os desenvolvedores cadastrados no ambiente, papéis desempenhados, projetos em execução e atividades já executadas, o próprio modelo de processo, e outras – são mantidas no repositório do ambiente (banco de dados) e podem ser recuperados pelo SGW (para manter a execução

e o monitoramento do processo) e pelas páginas de *script* (ASP, *javascript*) para a geração das páginas de acesso dos desenvolvedores ao ambiente.

Quando um projeto é iniciado por um *Gerente de Projeto* (este é um papel que pode ser desempenhado por um desenvolvedor [RAT 2001]), uma instância do modelo de processo, definido na linguagem de modelagem do SGW, é criada. A partir deste momento, qualquer modificação ocorrida dentro deste projeto (ao nível de atividades) é monitorada pelo SGW que, segundo o modelo de processo, toma as ações pertinentes em relação ao mesmo projeto. Modificação do estado de um projeto não causa modificações em um outro projeto, são instâncias totalmente separadas e que não influenciam umas as outras.

A arquitetura proposta neste capítulo servirá de base para a construção do protótipo deste trabalho, o qual é descrito no capítulo 6. A construção, e posterior avaliação, deste protótipo servirá como uma validação da arquitetura e também da abordagem de construção baseada em produtos COTS apresentadas.

5.9 Resumo do Capítulo

Neste capítulo foi proposta uma arquitetura para a construção de ambientes de desenvolvimento de software. Esta arquitetura esta baseada na integração e aplicação de ferramentas já disponíveis comercialmente (produtos COTS). Este tipo de abordagem visa diminuir o custo e tempo de construção de ambientes de desenvolvimento de software, melhorando a performance do ambiente por integrar ferramentas mais especializadas e proporcionar o reuso de ferramentas já existentes nas organizações Foram apresentados os benefícios e custos associados ao desenvolvimento com produtos COTS e como a utilização de SGW como um dos componentes de construção pode facilitar a construção de ADS-CP.

6 Protótipo Implementado

Neste capítulo será descrito o protótipo implementado durante este trabalho. Os componentes, interfaces, processos, itens de dados, pastas associadas, partes de códigos-fonte, etc. serão apresentados, mostrando-se os detalhes de implementação e configuração de ferramentas externas.

O protótipo implementado neste trabalho será nomeado **WOSDIE** (***WO****rkflow-based* ***S****oftware* ***D****evelopment* ***I****ntegrated* ***E****nvironment*). Assim, qualquer referência ao mesmo poderá ser feita utilizando este nome.

O WOSDIE foi implementado com base na arquitetura descrita no capítulo 5. O Sistema de Gerência de Workflow utilizado na implementação foi o *Microsoft Exchange 2000 Server* (E2K) [EXC 2001] [MIC 2001] [MIC 2001a]. O *Web Storage System* (WSS) foi utilizado como o banco de dados do ambiente. Algumas ferramentas utilizadas durante o processo de desenvolvimento de software foram integradas ao WOSDIE como: *MS Word*, *Ms Visual Basic*, *Rational Rose*, *Rational Test Manager*, etc. Pode-se integrar mais ferramentas ao ambiente sem maiores problemas. O protótipo foi integrado (software de integração, *glueware*) utilizando páginas ASP e funções em *Javascript* e *VBScript*; *VBScript* é a linguagem utilizada para a programação de condições e ações associadas dentro do E2K. A modelagem dos processos é feita utilizando o *Microsoft Workflow Designer for Exchange 2000 Server*, que é uma ferramenta de modelagem de processo de *workflow* que está integrada ao E2K. O papel de servidor de E-mail foi desempenhado pelo próprio E2K, visto que este é um servidor de colaboração que suporta um grande campo de atividades colaborativas [MIC 2000].

6.1 Configurações do *Microsoft Exchange 2000 Server*

A utilização do E2K depende da configuração de vários aspectos, incluindo a definição de pastas que irão ser utilizadas pelo WSS para o armazenamento de informações do E2K e de suas aplicações (WOSDIE).

Na seção 6.1.1 serão mostradas as pastas definidas dentro do E2K e que serão utilizadas pelo WSS para manter as informações, documentos e artefatos utilizados e produzidos pelo WOSDIE ou aplicativos relacionados.

6.1.1 Organização das Pastas Públicas Utilizadas pelo WOSDIE

O *Microsoft Exchange 2000 Server* (E2K) [MIC 2001] só pode ser instalado em um computador que possua como sistema operacional o *Windows 2000 Server* [MIC 2001], já que, como foi descrito na seção 3.6.6, o E2K está completamente integrado e utiliza vários serviços fornecidos pelo *Windows 2000 Server*. Após a instalação do E2K

em um computador, é criada uma unidade de disco virtual com o nome ¨Exchange¨, por exemplo: Exchange (M:). É nesta pasta que são criadas as sub-pastas das aplicações do E2K.

Para a implementação do WOSDIE foi criada uma hierarquia de pastas que mantêm os documentos associados e os dados gerados pelo mesmo. Estas pastas são as pastas do *Web Storage System*, descrito na seção 3.6.6.2, e que desempenham o papel de um repositório do ambiente. Esta hierarquia de pastas pode ser visualizada na Figura 6.1.

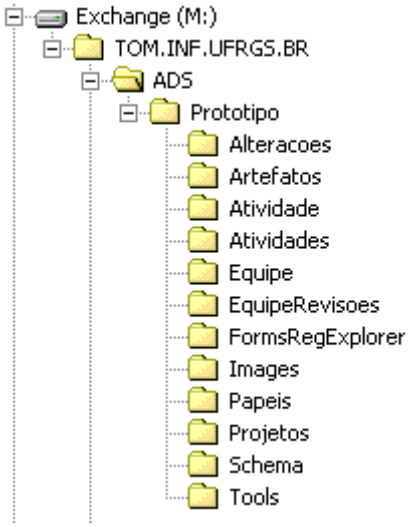

FIGURA 6.1 - Hierarquia de Pastas do WOSDIE

Como pode ser visualizado na Figura 6.1 existem várias pastas associadas ao WOSDIE. O propósito de cada pasta será descrito a seguir:

- Pasta “**Alteracoes**": esta pasta mantém os itens de dados referentes às alterações propostas durante um projeto de software. O *content class* desta pasta é “gpmgt:content-classes:Altfld”. Estes itens de alterações são criados quando algum novo requisito é definido pelos interessados (usuários), sendo um melhoramento do software; ou quando algum erro é encontrado durante os testes de sistema e componentes, sendo uma alteração provocada por erro de implementação.
- Pasta ”**Artefatos**”: está pasta mantém os artefatos que são enviados dos computadores clientes utilizados pelos desenvolvedores de um projeto para o computador servidor (operação de *upload*) que mantém a instalação do WOSDIE. O *content class* desta pasta é “urn:content-classes:folder”.
- Pasta “**Atividade**”: está pasta se encarrega de manter as definições e configurações das atividades que são modeladas no *Workflow Designer*. Cada atividade do modelo de processo (Figura 6.4) é configurada (papéis, ferramentas, gabaritos e manuais associados) e os dados referentes a estas atividades são mantidos nesta pasta. O *content class* desta pasta é “gpmgt:content-classes:ativambfld”.
- Pasta “**Atividades**”: nesta pasta são mantidos os dados referentes às instâncias de atividades que realmente foram realizadas durantes os projetos. O *content class* desta pasta é “gpmgt:content-classes:ativfld”.

- Pasta “**Equipe**”: os dados referentes a cada participante do grupo de desenvolvimento são mantidos nesta pasta. O *content class* desta pasta é “gpmgt:content-classes:equipefld”.
- Pasta “**EquipeRevisoes**”: todos os dados referentes as equipes de pessoas que realizam as revisões durante os projetos são armazenados nesta pasta. O *content class* desta pasta é “gpmgt:content-classes:equirevfld”.
- Pasta “**FormsRegExplorer**”: nesta pasta são armazenados os arquivos (documentos ASP e Entradas de Registro) utilizados para realizar o cadastro dos formulários (*forms*) das aplicações. Este cadastro é utilizado para a definição de quais documentos ASP, dependendo de qual pasta ou item (qual *content class* da pasta, ou do item) que está sendo acessado, devem ser interpretados pelo navegador do computador cliente. A Tabela 6.1 mostra os dados deste cadastro de registros de formulários para o WOSDIE. O *content class* da pasta “FormsRegExplorer” é “urn:content-classes:folder”. O cadastro destes formulários permite a gerência das interfaces de acesso do ambiente. Após este cadastro, de acordo com a URL que está sendo acessada, um documento ASP (ou mesmo HTML) é interpretado pelo navegador, mostrando as informações requisitadas de modo apropriado. O cadastro de formulários é feito acessando-se a seguinte URL: "http://nomeservidorexchange/nomepastaaplicacao/Schema", onde:
  - "nomeservidorexchange": é o nome do servidor que possui o Exchange 2000 Server instalado. No caso do WOSDIE é: "amadeus"
  - "nomepastaaplicacao": é o nome da pasta que mantém as subpastas da aplicação. No caso do WOSDIE é: "Prototipo". O endereço da interface de cadastro de formulários no WOSDIE é: http://amadeus/Prototipo/Schema.

  Para melhor ilustrar o funcionamento da pasta de registros de formulários, um exemplo será utilizado. Será utilizado como exemplo os dados da primeira linha da Tabela 6.1. Se for tentado acessar uma pasta ou objeto com o *content class* igual a gpmgt:content-classes:alteracao, não existir comando associado (*) e o modo de requisição for GET, será utilizado o documento AlteracaoEdit.asp, sem nenhum parâmetro de execução (*), para executar e mostrar as informações requisitadas.
- Pasta “**Images**”: esta pasta mantém as figuras utilizadas pelo WOSDIE. O *content class* desta pasta é “urn:content-classes:folder”.
- Pasta “**Papeis**”: esta pasta mantém os dados referentes aos papéis que podem ser desempenhados dentro do ambiente. Mantém os dados de nome do papel e uma *string* de identificação do papel. O *content class* desta pasta é “gpmgt:content-classes:papeisfld”.
- Pasta “**Projetos**”: todos os itens de projeto referentes aos projetos em execução ou já finalizados no WOSDIE são armazenados nesta pasta. É também para esta pasta que é definido o modelo de processo de software, definido através da ferramenta *Workflow Designer*. Os itens (itens de projeto) contidos nesta pasta vão passando de um estado para outro no decorrer da execução das atividades. Esta mudança de estados é definida de acordo com as condições e ações definidas no modelo de *workflow* modelado de acordo com o processo de software descrito na Figura 6.4. O *content class* desta pasta é “gpmgt:content-classes:projfld”.

- Pasta “**Schema**”: a pasta “Schema” mantém todos os documentos ASP utilizados nas interfaces de acesso do WOSDIE e os itens de registro de formulários (Tabela 6.1). O *content class* desta pasta é “urn:content-classes:schemafld”.
- Pasta “**Tools**”: nesta pasta são armazenadas as informações referentes às ferramentas cadastradas no ambiente; como o diretório do arquivo executável da ferramenta. O *content class* desta pasta é “gpmgt:content-classes:Toolsfld”.

TABELA 6.1 – *Explorer* de Registro de Formulários [MAR 2000] (Continua)
Registros de Formulários em *http://amadeus/Prototipo/Schema/*

Content Class	**Comando**	**Requisição**	**URL de Execução**	**Parâmetro de Execução**
gpmgt:content-classes:alteracao	*	GET	AlteracaoEdit.asp	*
gpmgt:content-classes:alteracao	save	POST	AlteracaoSave.asp	mode=edit
gpmgt:content-classes:alteracao	delete	POST	AlteracaoDelete.asp	*
gpmgt:content-classes:Altfld	*	GET	AlteracaoList.asp	*
gpmgt:content-classes:Altfld	addnew	GET	AlteracaoEdit.asp	*
gpmgt:content-classes:Altfld	save	POST	AlteracaoSave.asp	mode=addnew
gpmgt:content-classes:ativambfld	*	GET	ListarAtividade.asp	*
gpmgt:content-classes:ativambfld	save	POST	SalvarAtividade.asp	mode=addnew
gpmgt:content-classes:ativambfld	addnew	GET	ConfigurarAtividade.asp	*
gpmgt:content-classes:ativfld	*	GET	AtividadesList.asp	*
gpmgt:content-classes:atividade	*	GET	AtividadesEdit.asp	*
gpmgt:content-classes:atividade	saveativ	POST	AtividadesSave.asp	*
gpmgt:content-classes:atividade	revagenda	GET	RevisoesAgendaEdit.asp	*
gpmgt:content-classes:atividade	saveagenda	POST	RevisoesAgendaSave.asp	*
gpmgt:content-classes:atividade	revatualiza	GET	RevisoesAtualizaEdit.asp	*
gpmgt:content-classes:atividade	saveatualiza	POST	RevisoesAtualizaSave.asp	*
gpmgt:content-classes:atividade	revacomp	Get	RevisoesAcomp.asp	*
gpmgt:content-classes:atividadeamb	*	GET	ConfigurarAtividade.asp	*
gpmgt:content-classes:atividadeamb	save	POST	SalvarAtividade.asp	mode=edit
gpmgt:content-classes:atividadeamb	delete	POST	DeletarAtividade.asp	*
gpmgt:content-classes:equipe	*	GET	EquipeEdit.asp	*
gpmgt:content-classes:equipe	save	POST	EquipeSave.asp	mode=edit
gpmgt:content-classes:equipe	delete	POST	EquipeDelete.asp	*
gpmgt:content-classes:equipefld	addnew	GET	EquipeEdit.asp	*
gpmgt:content-classes:equipefld	*	GET	EquipeList.asp	*
gpmgt:content-classes:equipefld	save	POST	EquipeSave.asp	mode=addnew
gpmgt:content-classes:gpwebhome	*	GET	logon.asp	*
gpmgt:content-classes:gpwebhome	contents	GET	contents.asp	*
gpmgt:content-classes:gpwebhome	main	GET	main.asp	*
gpmgt:content-classes:gpwebhome	frameset	POST	Frameset.asp	*
gpmgt:content-classes:papeisfld	save	POST	PapeisSave.asp	mode=addnew
gpmgt:content-classes:papeisfld	*	GET	PapeisList.asp	*
gpmgt:content-classes:papeisfld	addnew	GET	PapeisEdit.asp	*
gpmgt:content-classes:papel	delete	POST	PapeisDelete.asp	*
gpmgt:content-classes:papel	save	POST	PapeisSave.asp	mode=edit

TABELA 6.1 – *Explorer* de Registro de Formulários [MAR 2000] (Continuação)
Registros de Formulários em *http://amadeus/Prototipo/Schema/*

gpmgt:content-classes:papel	*	GET	PapeisEdit.asp	*
gpmgt:content-classes:projeto	*	GET	PlanoProjetoEdit.asp	
gpmgt:content-classes:projeto	save	POST	PlanoProjetoSave.asp	mode=edit
gpmgt:content-classes:projeto	delete	POST	PlanoProjetoDelete.asp	*
gpmgt:content-classes:projeto	acompedit	GET	AcompProjetoEdit.asp	*
gpmgt:content-classes:projfld	*	GET	PlanoProjetoList.asp	*
gpmgt:content-classes:projfld	addnew	GET	PlanoProjetoEdit.asp	*
gpmgt:content-classes:projfld	acomp	GET	AcompProjetoList.asp	*
gpmgt:content-classes:projfld	save	POST	PlanoProjetoSave.asp	mode=addnew
gpmgt:content-classes:tool	*	GET	ToolEdit.asp	*
gpmgt:content-classes:tool	save	POST	ToolSave.asp	mode=edit
gpmgt:content-classes:tool	delete	POST	ToolDelete.asp	*
gpmgt:content-classes:Toolsfld	addnew	GET	ToolEdit.asp	*
gpmgt:content-classes:Toolsfld	*	GET	ToolList.asp	*
gpmgt:content-classes:Toolsfld	save	POST	ToolSave.asp	mode=addnew

6.2 Itens do Exchange 2000 e suas Respectivas Propriedades

Para a implementação do WOSDIE, foi necessária a criação de vários itens do *Exchange 2000 Server*. Estes itens são criados a partir da biblioteca CDO (*Collaboration Data Objects*) [MIC 2001a]. Os itens são descritos nas seções seguintes, onde as propriedades dos itens, a descrição destas propriedades e o nome da propriedade, ao nível de implementação, são apresentados. Os itens são utilizados para a criação de abstrações de conceitos fortemente associados a processos de software e ADS, como: atividades (incluindo revisões), trabalhadores (*workers*), papéis desempenhados, projetos de software, equipes de revisão, ferramentas utilizadas, solicitações de modificação de produtos de software e modelos de atividade.

6.2.1 Item Atividades

Os itens de atividades são armazenados na pasta “Atividades” (*gpmgt:content-classes:ativfld*). O *Content-Class* de uma pasta é definido no momento em que a pasta é criada. A propriedade *Content-Class* de uma pasta (ou qualquer outro objeto como documentos, itens do Exchange, programas, etc.) define o tipo de recurso que se esta acessando – para o caso de uma pasta, o seu *Content-Class* define quais tipos de itens são armazenados nesta pasta. A listagem da Figura 6.2 mostra um exemplo de código-fonte, escrito em *Visual Basic*, para a criação de uma pasta pública (no caso do exemplo, uma pasta denominada “Atividades”) e o seu *Content-Class* associado [MAR 2000].

```
' Extraído de [MAR 2000]
‘ Constantes das propriedades do Exchange 2000 Server
Const PROP_SCR As String = "urn:schemas-microsoft-com:exch-data:schema-collection-ref"
Const PROP_BASESCHEMA As String = "urn:schemas-microsoft-com:exch-data:baseschema"

Function CriaPastas()

  On Error GoTo ErroCriandoPastas

  Dim cnn As ADODB.Connection
  Dim rec As ADODB.Record
  Dim urlWSS As String
  Dim urlBaseSchema As String

  ' Constrói a URL para a aplicação Web Storage System (WSS)
  ‘ A Função GetStorageName retorna a URL do Servidor Exchange – por exemplo – http://Amadeus/
  urlWSS = GetStorageName & "ADS/"

  ' Constrói a URL para a aplicação WSS “non IPM schema folder”
  urlBaseSchema = urlWSS & "non_ipm_subtree/schema/"

  ' Abre a conexão para o WSS usando ADO
  Set cnn = New Connection
  With cnn
   .Provider = "exoledb.datasource"
   .Open urlWSS
   .BeginTrans
  End With

  Set rec = New ADODB.Record

  'Configuração da Pasta Pai
  With rec
   ‘ Abre a pasta Pai com alguns parâmetros para realizar configuração
   .Open "./Prototipo/", cnn, adModeReadWrite, adCreateCollection + adOpenIfExists

   ‘ Define o “ContentClass” da pasta
   .Fields("DAV:contentclass") = "gpmgt:content-classes:gpwebhome"
   .Fields(PROP_SCR) = "./schema/"  ‘ define a pasta que contem as configurações para exibição dos documentos
   .Fields.Update
   .Close
  End With

  ' Configuração da Pasta - Atividades
  With rec
   .Open "./Prototipo/Atividades/", cnn, adModeReadWrite, adCreateCollection

   ‘ Define o “ContentClass” da pasta pública
   .Fields("DAV:contentclass") = "gpmgt:content-classes:ativfld"
   .Fields(PROP_SCR) = "../schema/"
   .Fields.Update
   .Close
  End With
‘ Demais pastas podem ser criadas da mesma maneira aqui …

ErroCriandoPastas:
‘ Código de Erro aqui

End Function
```

FIGURA 6.2 - Exemplo de Código para a Criação de uma Pasta Pública

Na Tabela 6.2 o item “Atividades” é descrito, caracterizando cada uma de suas propriedades (ou atributos). As propriedades dos itens de “Atividades” são utilizadas para manter dados a respeito da execução das atividades dentro dos projetos. A cada início de uma atividade dentro de um projeto, um novo item “Atividades” é criado para manter os dados referentes a esta atividade.

TABELA 6.2 - Propriedades do Item de Atividades

Propriedade	Descrição	Nome da Propriedade
Itens Atividade – "content-class"	Itens armazenados na pasta "Atividades"	gpmgt:content-classes:atividade
Identificador do Projeto	String de identificação única de um projeto.	gpmgt:IdProjeto
Nome do Projeto	Nome dado ao projeto pelo gerente.	gpmgt:DescProjeto
Identificador da Atividade	Identificador da atividade, utilizado para fins de programação.	gpmgt:IdAtividade
Descrição da Atividade	String que descreve a atividade.	gpmgt:DescAtividade
Percentual Realizado	Percentual realizado da atividade.	gpmgt:Percentual
Responsável	Responsável pela execução da atividade.	gpmgt:Resp
Data Início	Data de início prevista para atividade.	gpmgt:DaInicio
Duração	Duração prevista da atividade.	gpmgt:Duração
Template	Template associado à atividade.	gpmgt:Template
Ferramenta	Ferramenta associada à atividade.	gpmgt:Ferramenta
Data Atual da Modificação	Data da realização da última modificação.	gpmgt:DaAtual
Observações	Observações gerais sobre a atividade.	gpmgt:Obs
Artefato	Artefatos já produzidos na realização da atividade.	gpmgt:Artefato gpmgt:Artefato1 gpmgt:Artefato2 ...
Status	Informa o estado de uma atividade de revisão. 1) Nova – a ser iniciada, 2) Agendada e 3) Concluída.	gpmgt:Status
Resultado	Informa o resultado de uma atividade de revisão.	gpmgt:Resultado
Hora da revisão	Hora da realização da atividade (revisão).	gpmgt:HoraRevisao
Duração Revisão	Duração, em horas, efetiva da atividade (revisão).	gpmgt:DuracaoRevisao
Atividade É Revisão?	*String* que define se uma atividade é ou não uma revisão.	gpmgt:ERevisao
Manual de Orientação	Manual de Orientação (*Work Guideline*) que descreve os passos da atividade	gpmgt:Guideline
Papel	Papel associado à atividade	gpmgt:Papel

6.2.2 Item Equipe

Os itens de Equipe são armazenados na pasta "Equipe" (*gpmgt:content-classes:equipefld*). Este item é descrito na Tabela 6.3.

TABELA 6.3 - Propriedades do Item de Equipe (Continua)

Propriedade	Descrição	Nome da Propriedade
Item Equipe – "content-class"	Itens armazenados na pasta "Equipe"	"gpmgt:content-classes:equipe"
Nome do trabalhador	Nome do trabalhador cadastrado.	gpmgt:NomeTrab
E-Mail do Trabalhador	Endereço de E-mail do trabalhador.	gpmgt:EmailTrab
Senha de Acesso	Senha fornecida pelo usuário no cadastro p/ futuro *logon* no sistema.	gpmgt:Senha
As propriedeades abaixo se referem aos possíveis papéis desempenhados pelas equipe cadastrada. Estes papéis são os que, atualmente, estão cadastrados no ambiente; novos papéis podem ser adicionados.		
Gerente de Projeto	Boleano que identifica se o trabalhador desempenha o papel de Gerente de Projeto.	gpmgt:idgerproj
Analista de Processos de Negócios	Idem para o papel de Analista de Processos de Negócios.	gpmgt:idanaprocneg
Projetista de Negócios	Idem para o papel de Projetista de Negócios.	gpmgt:idprojneg
Revisor de Modelos de Negócio	Idem para o papel de Revisor de Modelos de Negócio.	gpmgt:idrevmodneg
Analista de Sistemas	Idem para o papel de Analista de Sistemas.	gpmgt:idanasis
Especificador de Requisitos	Idem para o papel de Especificador de Requisitos.	gpmgt:idespreq
Arquiteto de Software	Idem para o papel de Arquiteto de Software.	gpmgt:idarqsof
Projetista	Idem para o papel de Projetista.	gpmgt:idproj
Projetista de Interface com o Usuário	Idem para o papel de Projetista de Interface com o Usuário.	gpmgt:idprojintusu
Implementador	Idem para o papel de Implementador.	gpmgt:idimp

TABELA 6.3 - Propriedades do Item de Equipe (Continuação)

Revisor de Projeto	Idem para o papel de Revisor de Projeto.	gpmgt":idrevproj
Gerente de Controle de Alterações	Idem para o papel de Gerente de Controle de Alterações.	gpmgt:idgerconalt
Integrador	Idem para o papel de Integrador.	gpmgt:idint
Projetista de Teste	Ide7m para o papel de Projetista de Teste.	gpmgt:idprojtes
Testador	Idem para o papel de Testador.	gpmgt:idtes
Gerente de Instalação	Idem para o papel de Gerente de Instalação.	gpmgt:idgerins
Escritor Técnico	Idem para o papel de Escritor Técnico.	gpmgt:idesctec

6.2.3 Item Projeto

Os itens de Projetos são armazenados na pasta “Projetos” (*gpmgt:content-classes:projfld*). A Tabela 6.4 mostra os identificadores de todas as atividades do processo de desenvolvimento do ambiente, juntamente com a descrição e os papéis responsáveis pelas respectivas atividades. Identificadores são utilizados para identificar uma determinada atividade ao nível de implementação.

TABELA 6.4 - Identificadores, Descrição e Papéis Associados às Atividades do Processo de Desenvolvimento (Continua)

Ident. da Atividade	**Descrição da Atividade**	**Papel Responsável pela Atividade**
CapturaVocabulário	Capturar um Vocabulário de Negócios Comum	Analista de Sistemas (*System Analyst*)
DefCasoUsoNegocio	Encontrar Atores e Casos de Uso de Negócio	Analista de Processos de Negócios (*Business Process Analyst*)
DetCUsoNegocio	Detalhar Casos de Uso de Negócio	Projetista de Negócios (*Business Designer*)
DefArqNegocio	Definir a Arquitetura de Negócio	Analista de Processos de Negócios (*Business Process Analyst*)
EncTrabEntNegocio	Encontrar Trabalhadores e Entidades de Negócio	Projetista de Negócios (*Business Designer*)
EliSolInt	Elicitar Solicitação dos Interessados	Analista de Sistemas (*System Analyst*)
DesVisao	Desenvolver a Visão	Analista de Sistemas (*System Analyst*)
DesCasoNegocio	Desenvolver o Caso de Negócio	Analista de Processos de Negócios (*Business Process Analyst*)
DesPlanoDesSoftware	Desenvolver o Plano de Desenvolvimento de Software	Gerente de Projeto (*Project Manager*)
DesPlanoIteracao	Desenvolver o Plano de Iteração	Gerente de Projeto (*Project Manager*)
EncAtoresCasosUso	Encontrar Atores e Casos de Uso	Analista de Sistemas (*System Analyst*)
DetCasosUso	Detalhar Casos de Uso	Especificador de Requisitos (*Requirements Specifier*)
PrioCasosUso	Priorizar Casos de Uso	Arquiteto de Software (*Software Architect*)
ModIntUsuario	Modelar a Interface com o Usuário	Projetista de Interface com o Usuário (*User Interface Designer*)
ProtIntUsuario	Prototipar a Interface com Usuário	Projetista de Interface com o Usuário (*User Interface Designer*)
AnaCasosUso	Analisar Casos de Uso	Projetista (*Designer*)
ProjCasosUso	Projetar Casos de Uso	Projetista (*Designer*)
ProjClasses	Projetar Classes	Projetista (*Designer*)
AnaArquitetura	Analisar a Arquitetura	Arquiteto de Software (*Software Architect*)
EstModImplementacao	Estruturar Modelo de Implementação	Arquiteto de Software (*Software Architect*)
ImpComponentes	Implementar os Componentes	Implementador (*Implementer*)
ExeTesUnidade	Executar Testes de Unidade	Implementador (*Implementer*)
IntComponentes	Integrar Componentes	Integrador (*Integrator*)
IntSistema	Integrar o Sistema	Integrador (*Integrator*)
PlanTeste	Planejar o Teste	Projetista de Testes (*Test Designer*)
ExecTeste	Executar o Teste	Testador (*Tester*)
DesPlaInstalacao	Desenvolver o Plano de Instalação	Gerente de Instalação (*Deployment Manager*)
DesArtInstalacao	Desenvolver os Artefatos de Instalação	Implementador (*Implementer*)
DesManUsuario	Desenvolver o Manual do Usuário	Escritor Técnico (*Technical Writer*)
FecFase	Fechar Fase	Gerente de Projeto (*Project Manager*)
FecProjeto	Fechar Projeto	Gerente de Projeto (*Project Manager*)
SubSolAlteracao	Submeter Solicitação de Alteração	Qualquer Desenvolvedor (*Any Role*)

TABELA 6.4 - Identificadores, Descrição e Papéis Associados às Atividades do Processo de Desenvolvimento (Continuação)

EliSolAltInteressados	Elicitar Solicitações de Alteração dos Interessados	Analista de Sistemas (*System Analyst*)
RevModObjNegocio	Revisar o Modelo de Objetos de Negócio	Revisor de Modelo de Negócios (*Business Model Reviewer*)
RevCasoNegocio	Revisar o Caso de Negócio	Revisor de Modelo de Negócios (*Business Model Reviewer*)
RevPlanoDesSoftware	Revisar o Plano de Desenvolvimento de Software	Revisor de Projeto (*Project Reviewer*)
RevPlanoIteração	Revisar o Plano de Iteração	Revisor de Projeto (*Project Reviewer*)
RevProjeto	Revisar o Projeto	Revisor de Projeto (*Project Reviewer*)
AvalTeste	Avaliar o Teste	Projetista de Testes (*Test Designer*)
RevGQS	Auditoria pelo Grupo de Garantia de Qualidade	Revisor de Projeto (*Project Reviewer*)
RevSolAltInteressados	Revisar Solicitações de Alteração dos Interessados	Gerente de Controle de Alterações (*Change Control Manager*)
RevRequisitos	Revisar Requisitos	Analista de Sistemas (*System Analyst*)
RevSolAlteracao	Revisar Solicitação de Alteração	Gerente de Controle de Alterações (*Change Control Manager*)

A Tabela 6.4 descreve as atividades que foram modeladas no *Workflow Designer* (Figura 3.7) para a experimentação do protótipo. O processo de software pode ser modificado, assim, a retirada, ou mesmo adição de novas atividades pode ser realizada normalmente, basta se incluir (ou retirar) as atividades na modelagem do processo (*Workflow Designer*) e cadastrar e configurar as mesmas nas interfaces de configuração de atividades do protótipo.

A Tabela 6.5 descreve as propriedades dos itens de “Projeto”. Um item de projeto armazena informações sobre todas as atividades do processo de desenvolvimento (citadas acima), assim, serão demonstradas (Tabela 6.5) somente as propriedades referentes a uma atividade ("Capturar um Vocabulário de Negócios Comum"), as propriedades das demais atividades também são utilizadas, mas não serão citadas aqui com intuito de diminuir o tamanho e complexidade da tabela. Estas propriedades possuem nomes (que podem ser definidos nas interfaces de configuração de atividades do protótipo) que associam a mesma a sua respectiva atividade, como é demonstrado com as propriedades referentes à atividade "Capturar um Vocabulário de Negócios Comum".

TABELA 6.5 - Propriedades do Item de Projeto

Propriedade	**Descrição da Propriedade**	**Nome da Propriedade**
Item de Projeto - (*content-class*)	É uma string que identifica unicamente cada projeto. São armazenados na pasta “Projetos”.	"gpmgt:content-classes:projeto"
Gerente do Projeto	Gerente responsável pelo projeto.	gpmgt:GerProjeto
Descrição	Nome que descreve o projeto.	gpmgt:DescProjeto
Iteração Inicial	Booleano que indica se o processo está ou não em sua iteração inicial.	gpmgt:IterIni
Opção de Início	Booleano que indica se o projeto é para ser iniciado imediatamente ou se é paras ser salvo como rascunho.	gpmgt:IterIni
Estado Anterior	*String* que indica qual era o estado anterior do projeto.	gpmgt:StateFrom
Abaixo estão as propriedades referentes à atividade "Capturar um Vocabulário de Negócios Comum".		
Responsável pela atividade "Capturar um Vocabulário de Negócios Comum"	È o responsável pela atividade. **Obs**: Todas as atividades restantes também são caracterizadas por propriedades como esta. As propriedades são nomeadas de acordo com os respectivos identificadores de atividades.	gpmgt:RespVocabulario
Data Início	Dada de Início desta atividade.	gpmgt:DaInicioVocabulario
Duração	Duração desta Atividade	gpmgt:DuracaoVocabulario

6.2.4 Item Equipe de Revisão

Os itens de Equipe de Revisão são armazenados na pasta “EquipeRevisoes” (*gpmgt:content-classes:equirevfld*). Este item é descrito na Tabela 6.6.

TABELA 6.6 - Propriedades do Item de Equipe de Revisão

Propriedade	**Descrição**	**Nome da Propriedade**
Item EquipeRevisão	São os itens contidos na pasta “EquipeRevisoes”	"gpmgt:content-classes:equiperevisao"
Identificador da Atividade de Revisão	String de identificação única de uma atividade de revisão.	gpmgt:IdRevisao
Identificador do Projeto	String de identificação única de um projeto.	gpmgt:IdProjeto
Identificador do Participante	String de identificação única de um trabalhador participante da revisão.	gpmgt:DescPartic

6.2.5 Item Solicitação de Alteração

Os itens de alterações são um tipo especial de artefatos. Estas mantêm os dados sobre as alterações propostas e/ou realizadas durante o desenvolvimento. Os itens de alteração são armazenados na pasta “Alteracoes” (*gpmgt:content-classes:altfld*). Este item é descrito na Tabela 6.7.

TABELA 6.7 - Propriedades do Item de Alteração

Propriedade	**Descrição**	**Nome da Propriedade**
Item Alteração – “content-class”	São os itens contidos na pasta “Alteracoes”.	"gpmgt:content-classes:alteracao"
Projeto	Identificador do Projeto a ser Modificado.	gpmgt:Projeto
Tipo Alteração	Define qual é o tipo de solicitação de alteração: **Melhoramento** – interessados solicitam a inclusão ou melhoramento de alguma funcionalidade ao produto; **Problema** – testadores encontrar algum erro no protótipo, descrevem o erro e possível resolução através de Solicitação de Alteração;	gpmgt:TipoSolAlt
Título da Alteração	Título que descreve sucintamente a alteração a ser realizada.	gpmgt:TituloAlt
Data da Submissão	Data em que a solicitação foi submetida.	gpmgt:DataSubAlt
Requisitante	È o usuário que fez a solicitação de alteração.	gpmgt:Requisitante
E-Mail do Requisitante	E-Mail do Requisitante da Alteração.	gpmgt:MailRequisitante
Prioridade	Prioridade para a realização da alteração.	gpmgt:Prioridade
Falha Crítica	Descreve a falha crítica ocasionada pelo erro, falha que ocasionou a descoberta do erro.	gpmgt:DescFalha
Descrição do Problema	Descreve o problema como um todo, descrevendo os módulos afetados pelo defeito e quais os problemas ocasionados por este(s) defeito(s).	gpmgt:DescProb
Motivo de Descontentamento	Motivo pelo qual o usuário resolveu pedir um melhoramento no produto.	gpmgt:DescMotivo
Descrição do Melhoramento	Descreve o que deve ser melhorado, delimitando escopo e definindo quais aspectos afetados por esta mudança.	gpmgt:DescMelhor
Descrição da Ação Proposta	Descreve a possível ação a ser tomada para resolução da Solicitação de Alteração.	gpmgt:DescAcaoProp

6.2.6 Item de Configuração de Atividade

Os itens de “configuração de atividade” mantêm os dados de configuração referentes às atividades modeladas no *Workflow Designer*. Estes dados definem as propriedades das atividades do modelo de processo; os dados referentes as atividades já

instânciadas são mantidos nos itens de “atividades”. As propriedades associadas ao item de “configuração de atividade” são descritos na Tabela 6.8.

TABELA 6.8 – Propriedades do Item de Configuração de Atividade

Propriedade	**Descrição**	**Nome da Propriedade**
Item de configuração de atividade – “content-class”	São os itens contidos na pasta “Atividade”.	“gpmgt:content-classes:atividadeamb”
Descrição da Atividade	*String* de descrição da atividade.	gpmgt:DescAtividade
É revisão	Booleano que define se a atividade é uma revisão	gpmgt:Erevisao
Ferramenta	Ferramenta associada à atividade.	gpmgt:Ferramenta
Manual de Orientação	Define qual é o manual de orientação para a execução da atividade	gpmgt:Guideline
Identificador da Atividade	*String* de identificação da atividade	gpmgt:IdAtividade
Papel	Papel associado à atividade	gpmgt:Papel
Gabarito	Gabarito associado à atividade (ao artefato da atividade)	gpmgt:Template

6.2.7 Item de Configuração de Ferramenta

Os itens de configuração de ferramenta mantêm os dados de configuração referente às ferramentas cadastradas no ambiente. Estes itens armazenam informações pertinentes das ferramentas de apoio ao desenvolvimento de software integradas ao WOSDIE. As propriedades deste item são mostradas na Tabela 6.9.

TABELA 6.9 - Propriedades do Item de Configuração de Ferramentas

Propriedade	**Descrição**	**Nome da Propriedade**
Item de configuração de ferramenta – “content-class”	São os itens contidos na pasta “Tools”.	“gpmgt:content-classes:tool”
Descrição da Ferramenta	Descrição da ferramenta	gpmgt:DescFerramenta
Diretório do Executável	É o caminho de diretório onde está o executável da ferramenta na máquina cliente.	gpmgt:Diretorio
IP do computador cliente	Endereço IP referente ao computador cliente que cadastrou a ferramenta.	gpmgt:IpUsuario
Nome da Ferramenta	Nome da Ferramenta	gpmgt:NomeFerramenta

6.2.8 Item de Configuração de Papéis

Os itens de configuração de papéis mantêm os dados de configuração referentes aos papéis que podem ser desempenhados pelos trabalhadores cadastrados no ambiente. As propriedades dos itens de configuração de papéis é descrito na Tabela 6.10.

TABELA 6.10 - Propriedades do Item de Configuração de Papéis

Propriedade	**Descrição**	**Nome da Propriedade**
Item de configuração de papéis – “content-class”	São os itens contidos na pasta “Papeis”.	“gpmgt:content-classes:papel”
Identificador do Papel	*String* de identificação do papel.	gpmgt:IdRole
Descrição do Papel	*String* de descrição do papel.	gpmgt:Role

6.2.9 Relacionamento entre os Itens do Exchange utilizados no WOSDIE

Na Figura 6.3 é mostrado o modelo de classes que descreve os relacionamentos entre os itens utilizados na implementação do WOSDIE.

Um item de *projeto* é composto pelas *atividades* executadas neste projeto, pelas *equipes de revisão* formadas durante o projeto e as *solicitações de alteração* criadas durante o projeto. Um item de *configuração de atividade* define as caracteríticas de um tipo de atividade e tem associado um item de *configuração de papéis* (que define qual papel é responsável por aquele tipo de *atividade*), um item de *configuração de ferramenta* (definindo a ferramenta utilizada para realização daquele tipo de atividade) e o item de *atividade* que é criado dinamicamente a cada início de nova atividade com base nas configurações definidas no item de *configuração de atividade*. Um item de *equipe* é criado a cada cadastro de novo trabalhador no WOSDIE. Durante a execução do processo, integrantes da equipe podem ser associados a uma revisão (um item de *Equipe Revisão* é criado para manter informações sobre a revisão e os seus participantes). Um membro da equipe, desempenhando o papel de Gerente de Projeto (gera a associação entre *Equipe* e *Projeto*), faz a atribuição de responsabilidade de atividades aos trabalhadores da equipe de acordo com os papéis desempenhados pelos mesmos, criando assim uma associação entre *Equipe* e *Atividade*. Itens de *Solicitação de Alteração* e *Equipe Revisão* estão associados à *Atividade* porque os mesmos armazenam informações a respeito de um tipo especial de atividade.

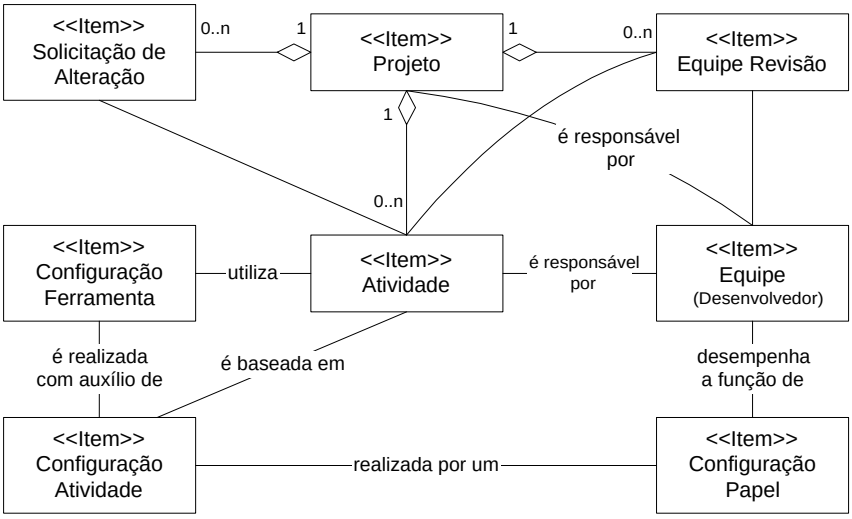

FIGURA 6.3 - Modelo de Classes Relacionando os Itens Utilizados no WOSDIE

6.3 Processo de Software do Ambiente

Como já definido anteriormente, o protótipo WOSDIE desenvolvido durante este trabalho deve proporcionar o controle do processo de desenvolvimento de software a ser seguido durante os projetos. Este controle é realizado pelo motor de *Workflow* do *Exchange 2000 Server*, que realiza o assinalamento de atividades aos responsáveis (os responsáveis por cada atividade do modelo de processo são definidos pelo *Gerente de*

Projeto em uma interface especial do WOSDIE), definição de quais atividades devem ser iniciadas após o término de outras, armazenamento dos dados sobre o *status* do projeto, etc. O processo de software utilizado pelo WOSDIE foi extraído do processo unificado da *Rational* (RUP) [RAT 2001] (Capítulo 4). Este processo abrange todos os *Workflows* descritos no RUP, mas que foram simplificados (pela retirada de algumas atividades) de modo a obter um processo completo, mas relativamente simples. Lembrando que o processo RUP é um processo adaptável, podendo ser adaptado de acordo com as necessidades dos usuários. Desta maneira, foram retiradas as atividades menos comuns (relacionadas apenas a alguns domínios de aplicação) aos processos de software. Também foram utilizados os gabaritos (*Templates*) e manuais de orientação (*Work Guidelines*) (Tabela 6.11) fornecidos pelo RUP através da disponibilização de links para os mesmos, e também links para toda a documentação do RUP [RAT 2001]. O processo de software modelado no ambiente (através do *Microsoft Workflow Designer*) é descrito graficamente, através do diagrama de atividades da notação UML [FOW 2000] [LAR 2000] [BOO 2000], na Figura 6.4.

Com base neste diagrama de atividades, o processo de desenvolvimento foi modelado no *Workflow Designer for Exchange 2000 Server*, que é a ferramenta gráfica para modelagem de *Workflows* que acompanha o *Exchange 2000 Server* e que já foi descrita na seção 3.6.6.1.

Durante a modelagem do processo de software utilizando a ferramenta *Workflow Designer* percebeu-se a necessidade da adaptação da notação do diagrama de atividades para a notação utilizada na ferramenta, que mais se assemelha a um diagrama de estados da UML. É necessário que as atividades sejam modeladas na ferramenta *Workflow Designer* como ações que fazem transições entre estados. Foi também necessário que alguns estados adicionais fossem criados com o propósito de melhorar e adaptar a modelagem do modelo de processo (Figura 6.4). Uma descrição de como modelar processos de software na ferramenta *Workflow Designer*, a partir de um diagrama de atividades, é feita na seção 6.3.1. Os estados do modelo de *Workflow* definem o estado em que determinado projeto (um projeto se refere aos artefatos, às informações e ao pessoal atrelados ao desenvolvimento de um novo produto de software) se encontra em algum momento da execução. No caso do WOSDIE, projeto também se refere a um item “projeto” definido a partir da biblioteca CDO. Desta maneira, a cada novo projeto, uma nova instância do modelo de processo de *Workflow* modelado na ferramenta é criada e, de acordo com o andamento do projeto, o item referente a este novo projeto vai passando de estado para estado, dependendo das condições alcançadas durante a execução das atividades. Um item de projeto representa, em nível conceitual, um projeto de software em andamento dentro do WOSDIE, mantendo informações sobre o projeto.

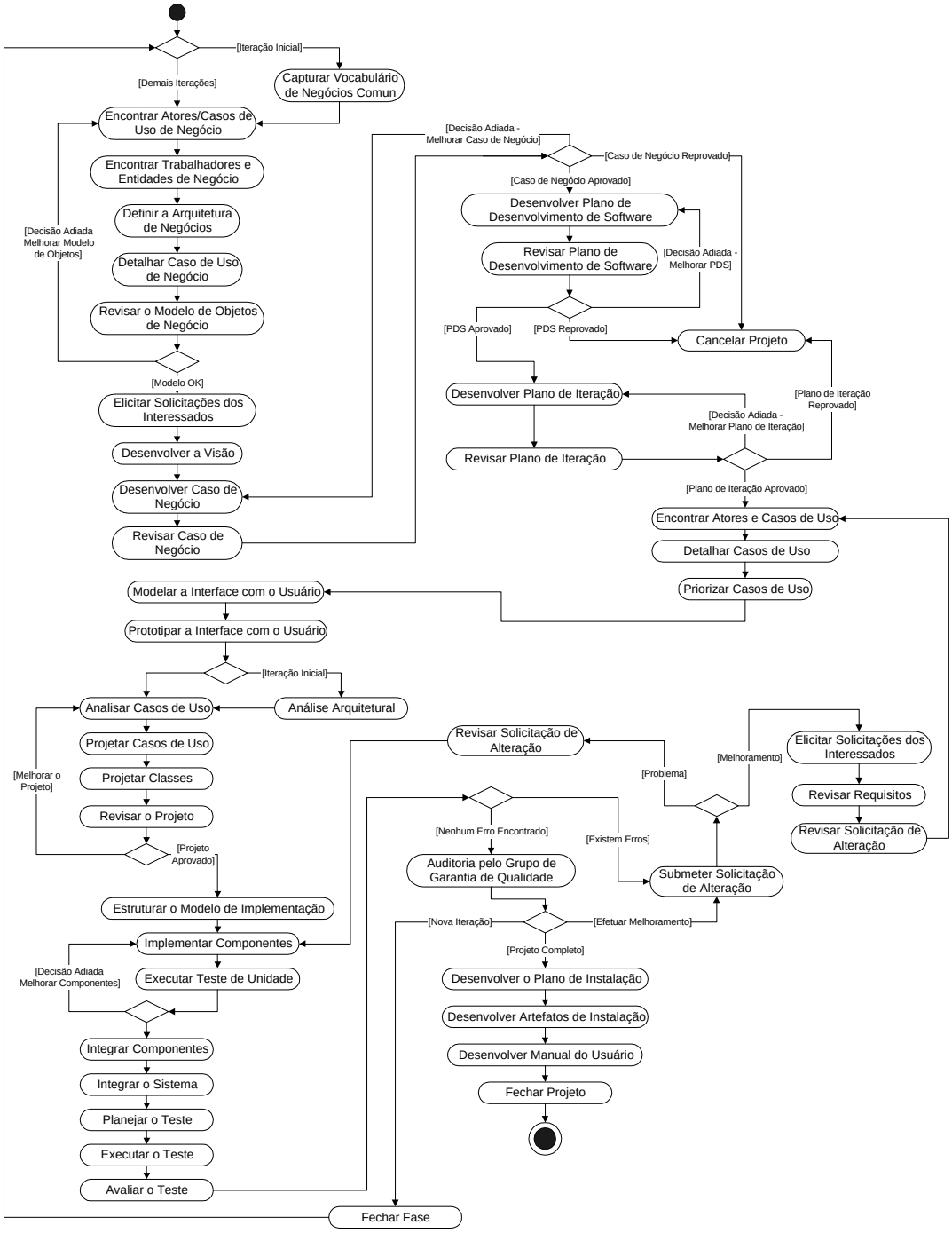

FIGURA 6.4 - Processo de Software do Ambiente.
Notação: Diagrama de Atividades. Extraído do Processo Unificado Rational [RAT 2001].

TABELA 6.11 - Atividades e seus Gabaritos e Manuais de Orientação Associados [RAT 2001]
Os modelos do Word e documentos HTML desta tabela são disponibilizados pelo RUP [RAT 2001]

Descrição da Atividade	**Gabarito (*Template*)**	**Manual de Orientação**
Capturar um Vocabulário de Negócios Comum	rup_bggloss.dot	ac_bccvo.htm
Encontrar Atores e Casos de Uso de Negócio	rup_sbs.dot	ac_fbauc.htm
Detalhar Casos de Uso de Negócio	rup_bucs.dot	ac_debuc.htm
Definir a Arquitetura de Negócio	rup_barchdoc.dot	ac_barc.htm
Encontrar Trabalhadores e Entidades de Negócio	rup_bucr.dot	ac_fbwke.htm
Elicitar Solicitação dos Interessados	rup_stkreq.dot	ac_elstk.htm
Desenvolver a Visão	rup_vision.dot	ac_dvisn.htm
Desenvolver o Caso de Negócio	rup_buscs.dot	ac_dbzcs.htm
Desenvolver o Plano de Desenvolvimento de Software	rup_sdpln.dot	ac_cosdp.htm
Desenvolver o Plano de Iteração	rup_itpln.dot	ac_ditpl.htm
Encontrar Atores e Casos de Uso	rup_ucspec.dot	ac_facuc.htm
Detalhar Casos de Uso	rup_ucspec.dot	ac_desuc.htm
Priorizar Casos de Uso	rup_sad.dot	ac_priuc.htm
Modelar a Interface com o Usuário		ac_uim.htm
Prototipar a Interface com Usuário		ac_uip.htm
Analisar Casos de Uso		ac_ucana.htm
Projetar Casos de Uso	rup_ucrs.dot	ac_ucdes.htm
Projetar Classes	rup_ucrs.dot	ac_cldes.htm
Analisar a Arquitetura	rup_sad.dot	ac_arcan.htm
Estruturar Modelo de Implementação	rup_sad.dot	ac_strim.htm
Implementar os Componentes		ac_impcl.htm
Executar Testes de Unidade	rup_sad.dot	ac_untst.htm
Integrar Componentes		ac_intsu.htm
Integrar o Sistema		ac_intsy.htm
Planejar o Teste	rup_tstpln.dot	ac_pltst.htm
Executar o Teste		ac_extst.htm
Desenvolver o Plano de Instalação	rup_dplpln.dot	ac_deppl.htm
Desenvolver os Artefatos de Instalação		ac_dinst.htm
Desenvolver o Manual do Usuário		ac_deusm.htm
Fechar Fase	rup_stass.dot	ac_pphco.htm
Fechar Projeto	rup_stass.dot	ac_pprco.htm
Submeter Solicitação de Alteração		ac_subcr.htm
Elicitar Solicitações de Alteração dos Interessados	rup_stkreq.dot	ac_elstk.htm
Revisar o Modelo de Objetos de Negócio		ac_rvbom.htm
Revisar o Caso de Negócio		ac_parev.htm
Revisar o Plano de Desenvolvimento de Software		ac_pprev.htm
Revisar o Plano de Iteração		ac_iprev.htm
Revisar o Projeto		ac_rvdes.htm
Avaliar o Teste		ac_evtst.htm
Auditoria pelo Grupo de Garantia de Qualidade		ac_prarv.htm
Revisar Solicitações de Alteração dos Interessados		ac_revcr.htm
Revisar Requisitos		ac_elstk.htm
Revisar Solicitação de Alteração		ac_revcr.htm

6.3.1 Modelando Processos de Software no *Microsoft Workflow Designer for Exchange 2000 Server*

Processos podem ser modelados na ferramenta *Workflow Designer*. A notação desta ferramenta se assemelha aos diagramas de estados da UML ([FOW 2000] [LAR 2000] [BOO 2000]).

Para a modelagem de um processo de software, normalmente é necessário que uma notação seja utilizada para descrever o modelo de processo que se deseja modelar em ferramentas de *workflow*. O processo modelado no WOSDIE é descrito na Figura 6.4. A notação utilizada para este modelo é o diagrama de atividades da UML.

Durante a modelagem foi necessário que as atividades descritas na Figura 6.4 fossem modeladas em termos de ações e estados, já que estes são os elementos principais da notação da *Workflow Designer*.

Para melhor descrever como foi modelado o modelo de processo na ferramenta *Workflow Designer*, os passos que foram seguidos, não necessariamente de forma seqüencial, serão descritos a seguir. Estes passos estão sujeitos a adaptações de acordo com o modelo de processo e a aplicação que se está construindo.

1. Para cada atividade no diagrama de atividades do modelo de processo da Figura 6.4, deve-se criar dois estados associados na notação da *Workflow Designer*.
    a) Um dos estados possuirá o nome da atividade, mas com o verbo que indica a ação da mesma no gerúndio, ou opcionalmente, uma sentença que dê idéia que a atividade está ocorrendo. Por exemplo: No caso de uma atividade denominada "Elicitar requisitos", cria-se um estado com o nome "Elicitando requisitos".
    b) O segundo estado deve ser criado com o nome que indique que a atividade foi concluída (com o verbo no particípio). Por exemplo: Para a atividade "Elicitar requisitos", cria-se um estado com o nome "Requisitos elicitados".
2. Cada estado criado a partir do item "1a" deve possuir no mínimo duas ações (da notação da *Workflow Designer*) chegando ao estado; uma para o caso em que a atividade que deve ser iniciada já possui um responsável definido, e outra, para o caso em que será necessário comunicar o *Gerente de Projeto* que uma atividade esta pendente de definição de responsável. Estas ações possuem o nome de uma atividade. Por exemplo: Para o estado “Elicitando requisitos”, criam-se ações com o nome “Elicitar Requisitos”. Este tipo de ação é a responsável pela criação de itens de “Atividades” por parte do *Exchange* (E2K) e faz com que as listas de atividades de cada usuário sejam atualizadas. As duas ações criadas a partir deste item (2) partem de estados (origem) que foram gerados tanto pelos itens “1a” quanto pelo item “1b”. As ações criadas neste item (item “2”) servem para a criação da mesma atividade dentro do processo de software, mas somente uma das ações será utilizada para instanciação da atividade durante a execução do processo, dependendo das condições alcançadas durante o projeto.
3. Cada estado criado a partir do item "1a" deve possuir no mínimo duas ações (da notação da *Workflow Designer*) saindo do estado.
    a) Uma das ações deve possuir o nome referente à próxima atividade. Se o processo possui a atividade “Elicitar Requisitos” e a atividade posterior a esta é a atividade “Desenvolver a Visão”, então se cria a ação com o nome “Desenvolver a Visão”, que passa para um estado criado a partir do item “1a” com base na atividade “Desenvolver a Visão”.
    b) A segunda ação deve possuir um nome que dê idéia que a atividade foi concluída. Por exemplo: Para o estado “Elicitando requisitos”, cria-se uma ação com o nome “Concluir elicitar requisitos”, que passa para um estado criado a partir do item “1b” (por exemplo: “Requisitos Elicitados”).
4. Normalmente alguns estados adicionais são necessários para ajustar a modelagem do processo. Por exemplo, na Figura 6.5 aparece um estado adicional no início do modelo de processo da *Workflow Designer*, o estado “Editando Projeto como Rascunho”. Este estado foi incluído porque nas interfaces de edição de projetos do protótipo (Figura 6.14) um projeto pode ser salvo como rascunho. Isso significa que um item de projeto pode ser criado, mas as atividades do projeto não são iniciadas,

pois o item de projeto permanece neste estado até que o usuário (geralmente um gerente de projeto) marque o projeto como pronto para iniciar.

Para ilustrar como é feita a modelagem de processos na ferramenta *Wokflow Designer*, é mostrado na Figura 6.5 a modelagem da parte inicial do modelo de processo do WOSDIE, definido na Figura 6.4, na notação da ferramenta *Workflow Designer*.

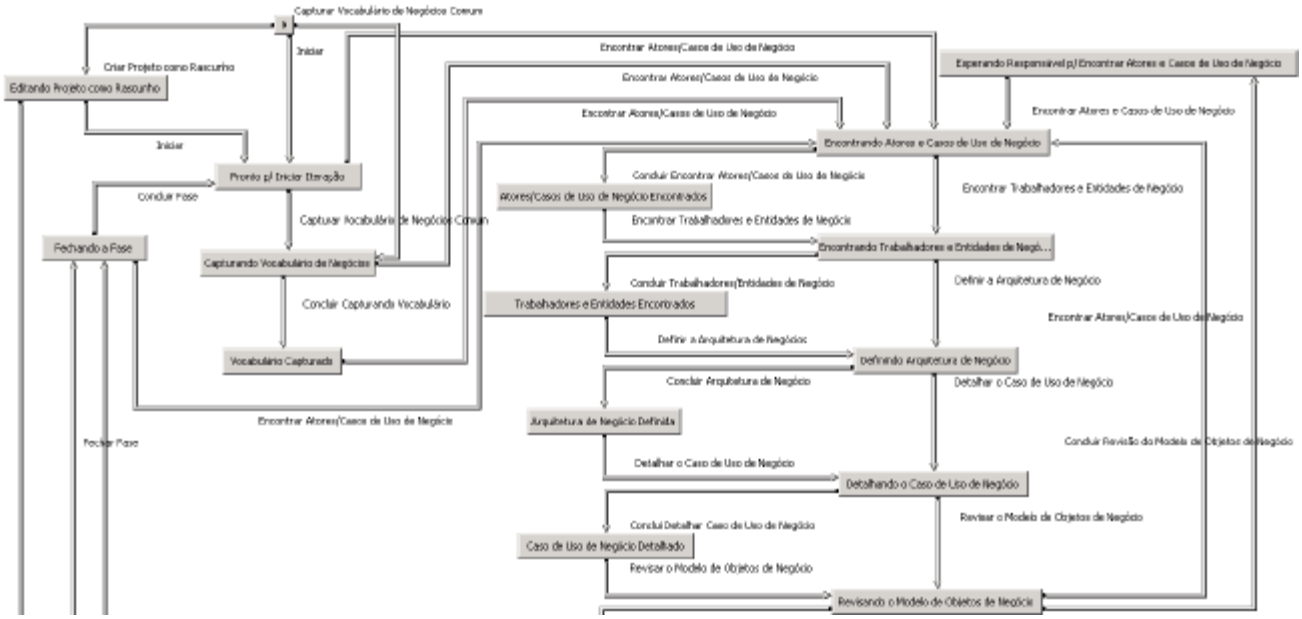

FIGURA 6.5 - Modelagem no Workflow Designer de parte do Modelo de Processo

A ferramenta *Workflow Designer* possui campos especiais para a programação de *scripts* (em *VBScript*). Estes campos são:

- Expressão de Condição (*Condition Expression*) – neste campo pode-se escrever a expressão lógica que deve ser verdadeira para que alguma ação seja realizada.
- Procedimento de *Script* de Ação (*Action Script Procedure*) – neste campo é programado o *script* da ação. Este *script* de ação só será executado se a expressão do campo de Condição desta ação for "Verdadeira" (*true*). Existem sete tipos de ações de *workflow*, que são as ações de:
  - Criar (*Create*) – este tipo de ação permite que um item seja criado e que passa para um determinado estado. Cada estado pode possuir somente uma ação Criar.
  - Deletar (*Delete*) – Este tipo de ação é utilizada quando um item esta em um estado qualquer do modelo de *workflow*, e a partir deste estado o item pode ser excluído.
  - Entrar (*Enter*) – Ação que é executada quando um item Entra em um novo estado (passa de um estado para outro).
  - Sair (*Exit*) – Tipo de Ação que é executada quando um item esta saindo de um estado e passando para outro.
  - Mudar (*Change*) – tipo de Ação que é executada quando um item é modificado. Dependendo da expressão de condição desta Ação e do modelo

de *workflow*, uma Ação de Mudar pode levar um item para um novo estado ou mesmo fazer com que permaneça no mesmo estado.

- Receber (*Receive*) – este tipo de ação permite que o *workflow* responda mensagens de E-mail enviadas para uma pasta publica habilitada para receber correio eletrônico.
- Expirar (*Expiry*) – este tipo de ação é utilizado quando um item pode ficar um determinado tempo em um estado, se este tempo for expirado, um *script* de ação deverá ser executado.

➢ Ações para: "Nome do Estado" (*Actions for:*) – neste campo é definido a ordem de avaliação das expressões de condição para as ações de um determinado estado. As ações que causam a troca de estados de um item devem ser avaliadas primeiramente.

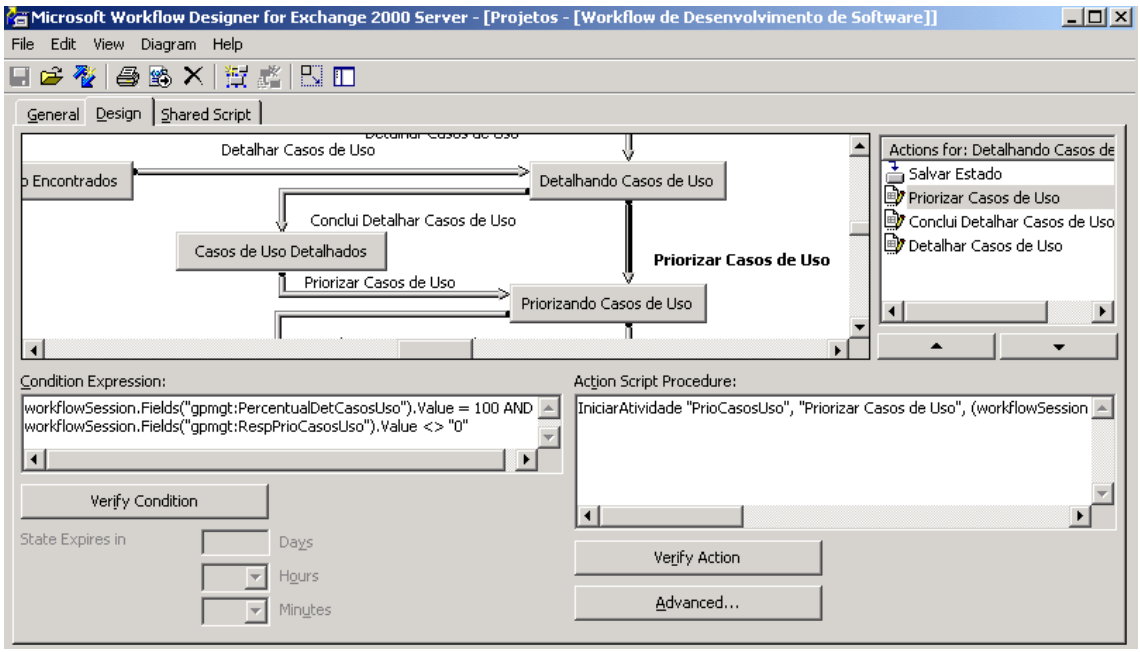

FIGURA 6.6 – Campos para Modelagem no *Workflow Designer*

Na Figura 6.6 pode ser observado como são construídas as expressões de condições, os *scripts* de ação e a ordem de avaliação das ações de um estado. A Figura 6.6 mostra o campo "*Actions for:*" com as ações associadas ao estado "Detalhando Casos de Uso". Dentre estas ações temos: a ação "Salvar Estado", que é uma ação do tipo "Entrar". As demais ações são do tipo "Mudar", sendo que as ações "Priorizar Casos de Uso" e "Conclui Detalhar Casos de Uso" são ações que causam transição entre estados (aparecendo uma seta para cada ação no modelo). A ação "Detalhar Casos de Uso" serve para captar um evento de modificação no item, mas que não causa uma transição de estados, sendo assim, os estados origem e destino para esta ação são os mesmos (não aparecendo assim uma seta associada a esta ação no modelo).

No campo "*Condition Expression*" pode ser observado a expressão de condição que deve ser avaliada para que o *script* da ação "Priorizar Casos de Uso" possa ser executado. Nesta expressão de condição é avaliado se o percentual executado da atividade anterior é igual a 100% e se a atividade "Priorizar Casos de Uso" já possui um responsável.

No caso em que a avaliação da expressão de condição é verdadeira, o *script* definido no campo "*Action Script Procedure*" será executado; como mostrado na Figura 6.6 será executado uma subrotina chamada "IniciarAtividade". Esta subrotina está definida no campo "*Shared Script*". O campo "*Shared Script*" permite a construção de subrotinas e funções escritas em *VBScript* que podem ser "chamadas" de qualquer ponto dentro dos campos de modelagem do *Workflow Designer*.

6.4 Ativação de Ferramentas a Partir do WOSDIE

As ferramentas de auxílio ao desenvolvimento das atividades de um projeto podem ser ativadas a partir das interfaces do WOSDIE. Para fornecer esta funcionalidade, foi utilizado o servidor de automação do sistema operacional *Windows*, através da Automação OLE (*Object Linking and Embedding* - Vinculação e Incorporação de Objetos) [OLE 2002] [MIC 2001a] [WHI 2002] [WHI 2002a].

Automação OLE permite que aplicações se comuniquem, troquem dados e controlem uma a outra. Permite que uma aplicação cliente crie e controle um objeto, utilizando a interface oferecida pelo mesmo [OLE 2002].

No WOSDIE a ativação de ferramentas externas se dá através de *scripts*, escritos em *JavaScript* e que rodam no computador cliente, que estão contidos nos documentos ASP que fornecem as interfaces de acesso do protótipo.

```
function startRose() {
	var rose = new ActiveXObject("Rose.Application");
	if (rose != null) {
		rose.Visible = true;
	}
}
```

FIGURA 6.7 - Função JavaScript para Ativação do Rational Rose

Na Figura 6.7 um exemplo de função de ativação de uma ferramenta externa (*Rational Rose*) é mostrada. A utilização do servidor de automação do *windows* é realizada através da instanciação de um novo objeto `ActiveXObject`, passando como parâmetro a *string* `"Rose.Application"`. No computador cliente, quando este *script* for executado, o servidor de automação vai procurar entre seus registros de ferramentas instaladas aquela que forneça a interface de instanciação `"Rose.Application"`, após isto, irá executar a ferramenta associada, que no caso exemplo é a ferramenta de edição de diagramas *Rational Rose*.

O WOSDIE fornece para cada tipo de atividade, quando necessário, uma ferramenta padrão para o auxílio a execução desta atividade. Para os casos em que os usuários (desenvolvedores) do WOSDIE quiserem utilizar outras ferramentas, sem ser as já disponibilizadas, é necessário que esta ferramenta seja cadastrada (fornecendo informações como nome da ferramenta, caminho do diretório do executável desta ferramenta no computador cliente, etc.) no ambiente. Após este cadastro, deverá também ser configurado no ambiente, as atividades que deverão ser realizadas com o auxílio desta nova ferramenta, assim, após este serviço de configurações, sempre que as

atividades associadas a nova ferramenta cadastrada aparecerem na lista de tarefas do usuário, um *link* de ativação para a nova ferramenta será adicionado ao lado do *link* de ativação da ferramenta padrão associada à atividade.

A ativação de ferramentas externas, que não são fornecidas de forma padrão pelo ambiente e que são cadastradas através de interfaces de configuração de ferramentas, se dá através da utilização um controle *ActiveX* chamado **LaunchinIE** [WHI 2002a] disponível gratuitamente pela *Internet* e que deve ser instalado no computador cliente que acessa as páginas do servidor que "hospeda" o WOSDIE. Este controle *ActiveX* permite que, através de um *link* para uma função *JavaScript* (Figura 6.8) que tem como parâmetro o caminho de diretório de um arquivo executável (ou mesmo um documento que possui uma aplicação associada para edição do mesmo) no computador local (computador cliente), aplicações externas sejam ativadas no computador cliente via *navegador*. Exemplos de *links* para as funções *JavaScript* citadas neste parágrafo são as seguintes.

(1)

```
<A href = "javascript:launchApp('C:\\Arquivos de Programas\\Microsoft
Office\\Office\\WINWORD.EXE')"> Ativar Microsoft Word! </A>)
```

(2)

```
<A href = "javascript:openDoc('C:\\Documentos\\Documento.doc')">
Abrir Documento.doc! </A>)
```

No primeiro caso, é passado para a função launchApp() (Figura 6.8) o caminho de diretório do arquivo executável da ferramenta. Neste caso, o controle *ActiveX* irá executar a ferramenta a partir do próprio executável. No segundo caso, é passado como parâmetro para a função openDoc() (Figura 6.8) o caminho de diretório de um documento ou um arquivo executável; a partir deste documento o controle *ActiveX* irá procurar (dentre as ferramentas instaladas no computador cliente) qual ferramenta está associada ao documento e executá-la; se for o caso de um arquivo executável, este será executado diretamente.

```
<SCRIPT LANGUAGE="javascript">
<!--
function launchApp(strCmdLine) {
      var obj = new ActiveXObject("LaunchingIE.Launch");
      obj.LaunchApplication (strCmdLine)
}
function openDoc(strDoc) {
      var obj = new ActiveXObject("LaunchinIE.Launch");
      obj.ShellExecute("open", strDoc);
}
//-->
</SCRIPT>
```

FIGURA 6.8 - Funções *JavaScript* utilizando o Controle *ActiveX* LaunchinIE [WHI 2002a]

Uma restrição em relação ao cadastro de novas ferramentas no ambiente, é a necessidade destas ferramentas gerarem arquivos compatíveis com os arquivos gerados pelas ferramentas padrão que já estão integradas ao WOSDIE. Isto é necessário para que os artefatos produzidos pelos diversos desenvolvedores associados aos projetos possam

ser analisados e editados por qualquer desenvolvedor, sem acontecer incompatibilidades de tipos de arquivos e ferramentas.

Uma outra restrição surge em relação à utilização de automação OLE e controles *ActiveX*; para que estas funcionalidades possam ser utilizadas através de navegadores *Web* na *Internet*, é necessário modificar as configurações de segurança dos navegadores cliente (item de menu "Ferramentas", sub menu "Opções da Internet"), modificando a configuração do item "*Inicializar e Executar Scripts de Controles ActiveX não Marcados como Seguros*" de *"Desativar"* para *"Confirmar"*. Em relação a utilização dentro de *Intranets*, pode-se considerar adicionar os *sites* da *Intranet* como *Sites Confiáveis* [WHI 2002] [OLE 2002].

Para a instalação do controle *ActiveX* **LaunchinIE** é necessário possuir o arquivo LaunchinIE.DLL (que pode ser "baixado" da *Internet* [WHI 2002a]); este arquivo pode ser instalado em qualquer pasta do computador cliente (controles *ActiveX* normalmente são colocados na pasta SYSTEM32 localizada no diretório de intalação do *Windows*). Após a copia do arquivo LaunchinIE.DLL para o diretório de escolha é necessário que este arquivo seja registrado no sistema *Windows*. Este registro pode ser realizado utilizando-se a aplicação de *console* REGSVR32.EXE, que normalmente está localizada na pasta SYSTEM do diretório de intalação do *Windows*, passando-se o nome do arquivo de instalação (LaunchinIE.DLL) como parâmetro. Depois de registrado o controle *ActiveX*, é necessário que seja criada a chave "HKEY_LOCAL_MACHINE/SOFTWARE/RockinFewl/LaunchinIE/Approved" no editor de registros do *Windows*. Nesta chave são definidos as URLs que podem utilizar os serviços do LaunchinIE, isto é feito nomeando os valores da chave com `url1`, `url2`, ... e com `- start` para a `url1`, conforme a Figura 6.9.

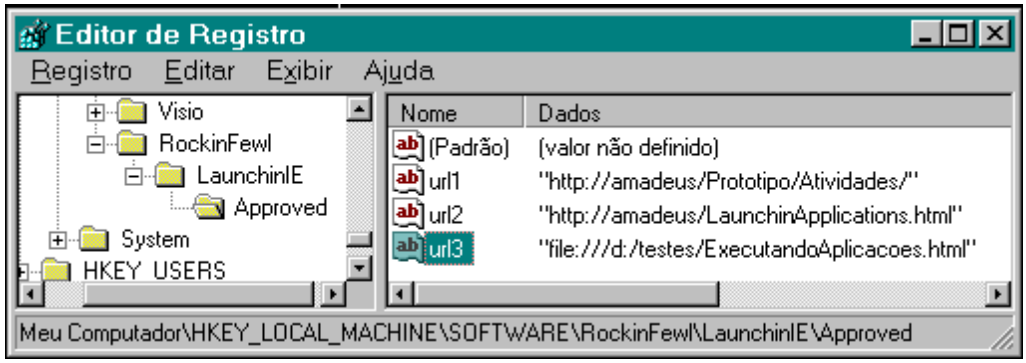

FIGURA 6.9 - Registrando URLs seguras para o Controle *ActiveX* LaunchinIE [WHI 2002a]

6.5 Artefatos Gerados

Durante o processo de desenvolvimento de software muitos tipos diferentes de documentos são gerados; estes são usualmente criados como parte de um grupo de trabalho e muitos projetistas compartilham os mesmos [OIN 99].

Os artefatos gerados pelos desenvolvedores durante o processo, podem ser armazenados em um repositório central, localizado no servidor. Esta funcionalidade

pode ser implementada através de operações de *upload*. Este tipo de operação permite que documentos localizados nos computadores clientes possam ser "carregados" para o servidor e armazenados no mesmo.

Para que uma operação de *upload* possa ser realizada, é necessário que seja instalado no servidor um componente de software. Vários componentes que fornecem funcionalidades de *upload* estão disponíveis gratuitamente na *Web*. No WOSDIE utilizou-se o componente de *upload* da Dundas Software [DUN 2002]. Este componente é disponibilizado em forma de um programa de instalação, que também acompanha exemplos de aplicação do componente e toda a documentação sobre a utilização do mesmo.

Cada projeto de software cadastrado no WOSDIE pode possuir vários artefatos associados. Durante o andamento de um projeto os desenvolvedores podem necessitar consultar artefatos já construídos anteriormente. Os artefatos de um projeto são disponibilizados através da geração dinâmica de um documento HTML que possui *links* para estes artefatos. Os *links* associados a cada artefato de software ativam, quando necessário, ferramentas externas associadas ao tipo de documento referente ao artefato.

Um problema ainda em aberto no WOSDIE é a integração dos artefatos. Diferentes ferramentas geram os artefatos e, por conseqüência, são gerados diferentes formatos de documentos. Esta possível incompatibilidade de documentos poderia ser resolvida se as ferramentas utilizadas no WOSDIE (cadastradas no WOSDIE) tivessem opções de geração de documentos em um formato padrão. Um formato que poderia ser utilizado seria o de documentos XML [XML 2001].

6.6 Arquitetura do Protótipo

No capítulo 5 foi descrita uma arquitetura de Ambientes de Engenharia de Software baseada em ferramentas comerciais como sistemas de gerência de *workflow* (SGW) e ferramentas de desenvolvimento de software. A *Internet* também faz parte da arquitetura, desempenhando o papel de plataforma base de funcionamento.

Com base na Figura 5.3, onde um modelo para a arquitetura proposta na Figura 5.2 é descrito, foi construído um diagrama de descrição de arquitetura (com base nos componentes, classes, documentos, etc., utilizados no WOSDIE) que descreve e modela o WOSDIE. Esta descrição da arquitetura visa mostrar os componentes utilizados na implementação do WOSDIE e como estes componentes interagem. O diagrama de descrição da arquitetura do WOSDIE é mostrado na Figura 6.10.

Em comparação ao modelo da Figura 5.3, o modelo da Figura 6.10 possui alguns elementos extras. Como foram utilizados itens do *Exchange 2000 Server* para a construção do WOSDIE, o modelo mostra estes itens associados aos componentes executáveis [LIM 2000] "*Documentos ASP***"** (por questões de simplicidade do modelo, não foram especificados cada um dos documentos ASP que gerenciam os itens do WOSDIE – alguns destes documentos possuem também funções implementadas em *JavaScript*), que por sua vez acessam e modificam as propriedades destes itens de acordo com o andamento dos projetos. Alguns *componentes executáveis* (*Contents.asp*, *Main.asp*, *Logon.asp*, *PaginaErro.asp* e *FrameSet.asp*) [LIM 2000], responsáveis pelo

controle de acesso e de interfaces do WOSDIE, também são mostrados. O *documento* [LIM 2000] "*Documentos HTML*" e suas especializações são arquivos no formato HTML e *Microsoft Word* (*Gabaritos RUP*) que são utilizados pelos desenvolvedores na execução de suas atividades e foram extraídos do RUP [RAT 2001]; estes documentos estão integrados ao WOSDIE. A modelagem dos elementos que tratam da parte de *internet* do WOSDIE foi realizada baseada no método WIDe, que é uma extensão à UML para Modelagem de Sistemas de Informação na Internet Baseados em Documentos, proposto por Lima [LIM 2000]. O componente "*ActiveX Dundas.UpLoad*" foi adicionado ao modelo para possibilitar as operações de *UpLoad* no servidor do WOSDIE; e os componentes "*ActiveX LaunchIE*" e "*Servidor de Automação do Windows*" para tratarem da ativação de ferramentas externas a partir do navegador *Web*.

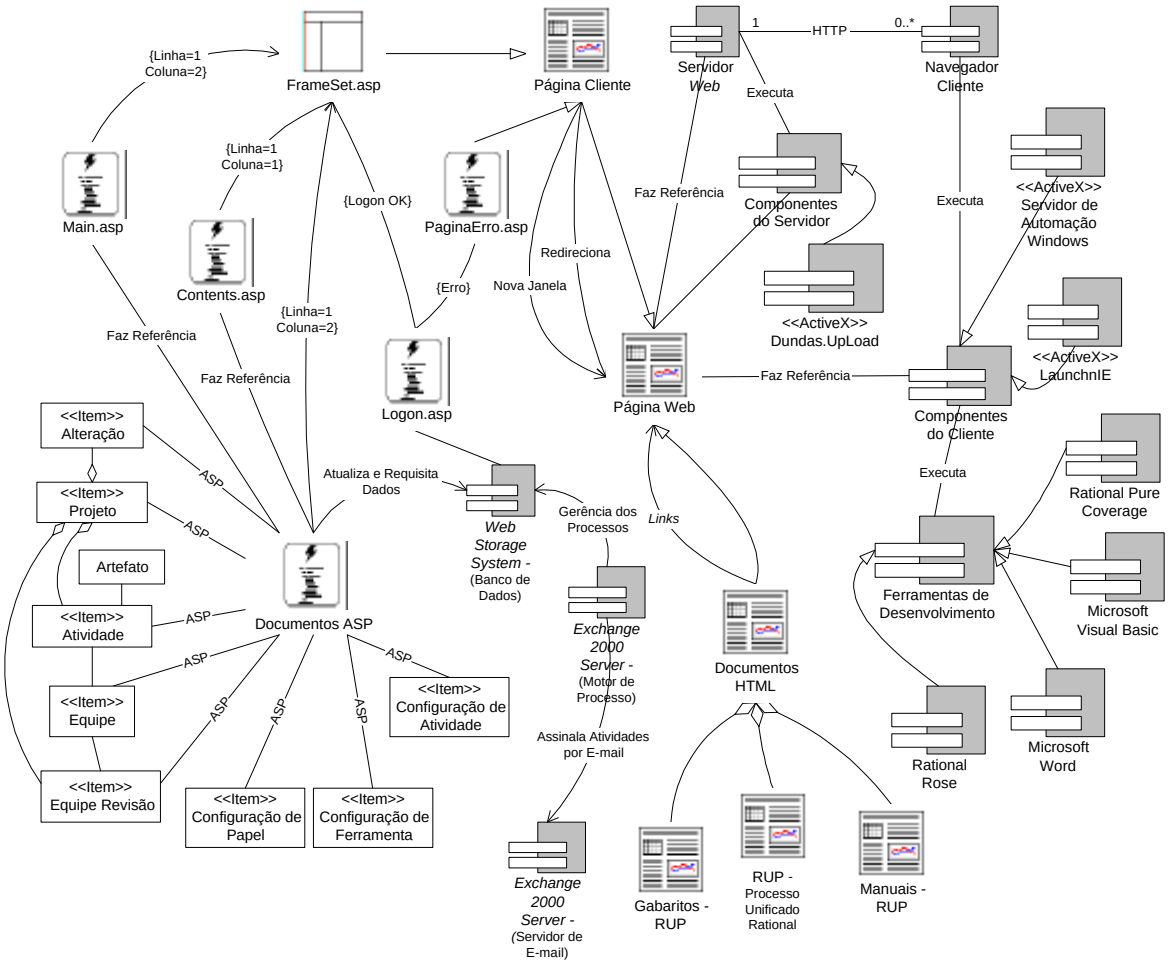

FIGURA 6.10 - Modelo de Componentes do WOSDIE

Como pode ser observado na Figura 6.10, existem algumas "Ferramentas de Desenvolvimento" associadas ao WOSDIE. Na realidade, outras ferramentas podem ser associadas, para isso, basta que as mesmas sejam cadastradas nas interfaces de cadastro de ferramentas. A utilização das quatro ferramentas em específico apresentadas na Figura 6.10 tem apenas o intuito de exemplificar possíveis ferramentas, e não citar todas as ferramentas que poderiam ser utilizadas no WOSDIE; não foram exemplificadas mais ferramentas para simplificar o diagrama.

6.7 Interfaces de Acesso do WOSDIE

O WOSDIE possui várias interfaces de acesso para seus usuários. Nesta seção estas interfaces serão apresentadas.

***Logon* no Ambiente**: para se acessar o ambiente é necessário que um *logon* seja executado pelo usuário. Cada pessoa cadastrada no ambiente possui um nome de usuário e uma senha para a realização do *logon*. A interface de *logon* no ambiente pode ser visualizada na Figura 6.11.

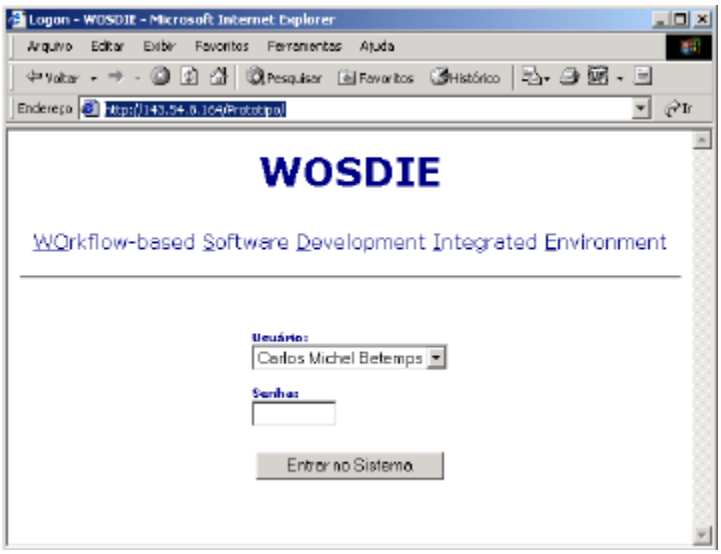

FIGURA 6.11 - Interface de *Logon* no Ambiente

Interface Principal: Após o *logon* do usuário no ambiente, a interface que é disponibilizada ao usuário é a demonstrada na Figura 6.12. Esta interface principal (*home* - em páginas *Web*) possui vários *links* que levam para outras interfaces e que disponibilizam as várias funcionalidades do ambiente através de outros *links*.

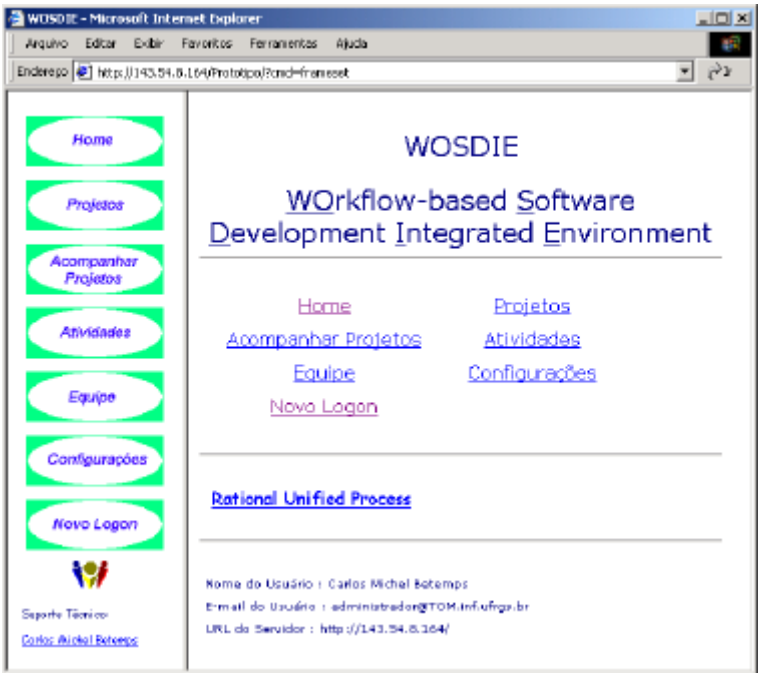

FIGURA 6.12 - Interface Principal do Ambiente

Projetos: quando um usuário clica no *link* "Projetos", a interface mostrada na Figura 6.13 aparece. A partir dos *links* contidos nesta página pode-se criar um novo projeto de software através do clique no botão "Adicionar Novo Projeto" ou pode-se editar projetos já existentes através de *links* que identificam os projetos individualmente. Quando qualquer um destes *links* e clicado, a interface da Figura 6.14 é mostrada. Esta interface permite a edição dos projetos, possibilitando a definição e edição de: nome do projeto, gerente responsável pelo projeto, os responsáveis por cada atividade e as dadas previstas para a execução e tempo estimado de duração de cada atividade do projeto.

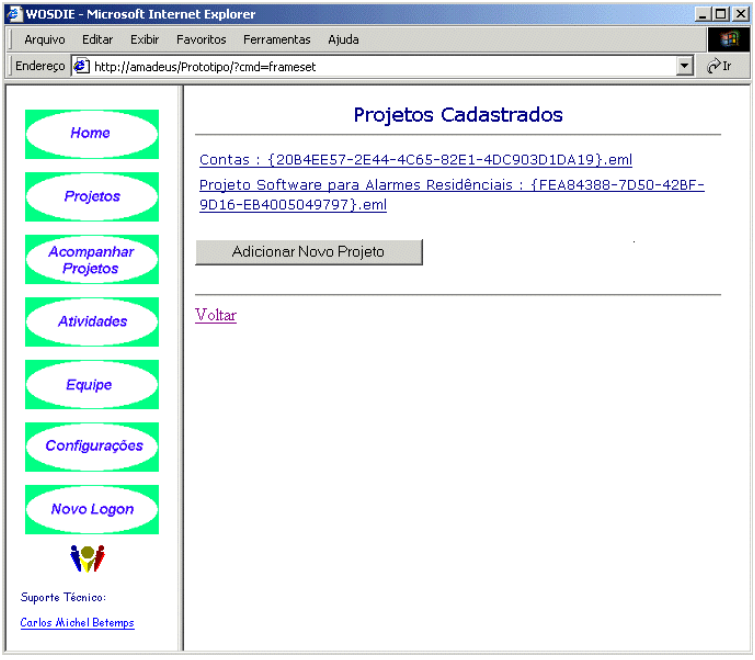

FIGURA 6.13 - Interface de Projetos Cadastrados

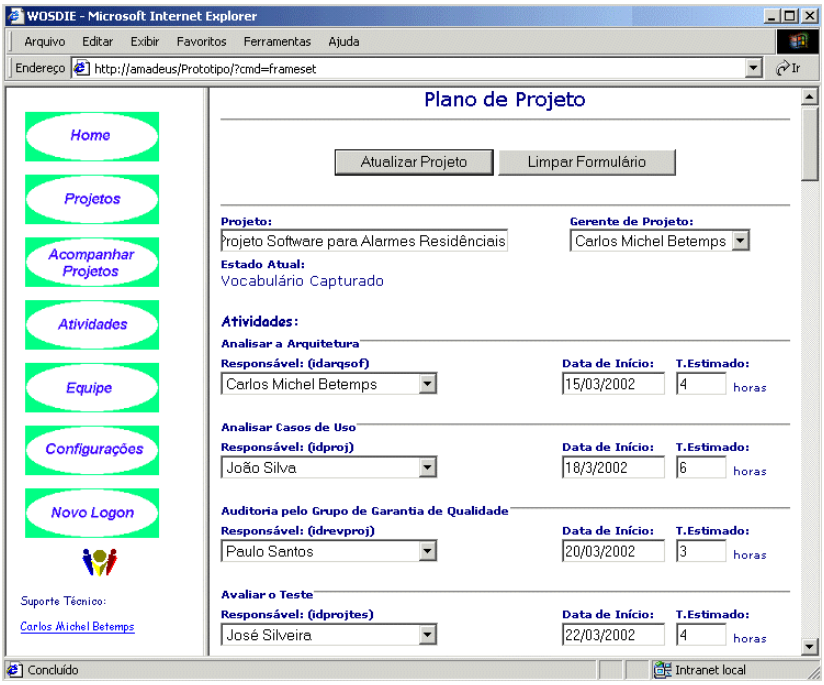

FIGURA 6.14 - Edição dos Planos de Projeto

Acompanhar Projetos: quando um usuário (normalmente os gerentes de projeto) precisa saber a situação dos projetos em execução (ou mesmo quiser avaliar projetos já concluídos) este pode clicar no *link* "Acompanhar Projetos". A partir deste *link* a interface da Figura 6.15 aparece, oferecendo *links* para a interface de acompanhamento dos projetos (Figura 6.16) e de acesso aos artefatos já concluídos dos projetos (Figura 6.17).

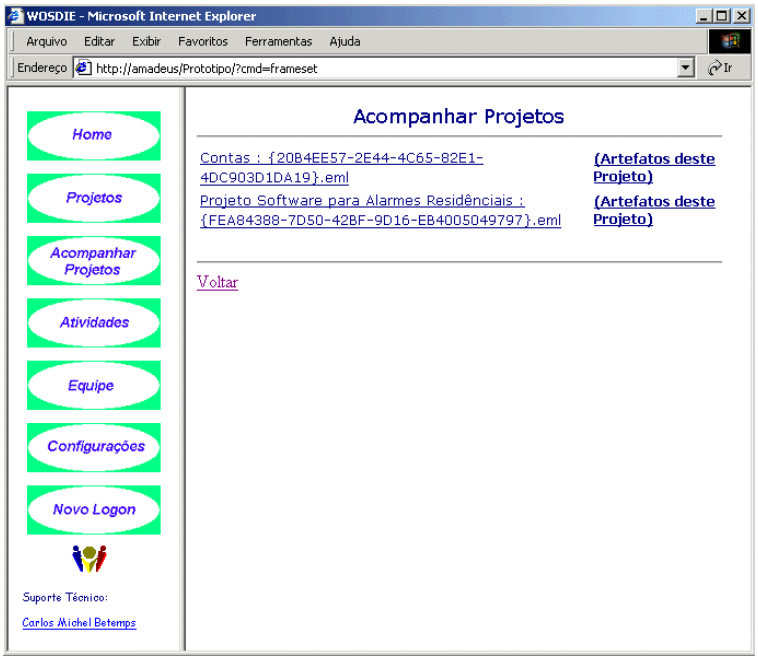

FIGURA 6.15 - Interface de Acompanhamento de Projetos e Integração de Artefatos

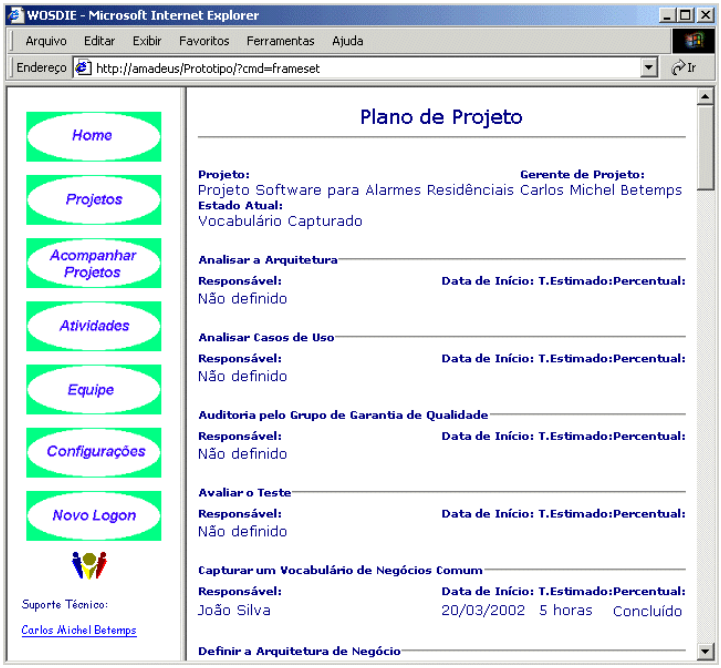

FIGURA 6.16 - Interface de Acompanhamento de um Projeto Individual

Atividades: Nos links "Atividades" da Figura 6.12 se obtém o acesso as listas individuais de trabalho (listas de atividades - worklists). Quando o usuário se loga no WOSDIE, o link "Atividades" é criado de acordo com este usuário, de modo que somente as atividades assinaladas a este usuário em específico serão mostradas. A Figura 6.18 mostra a "Lista de Atividades" associada a um usuário. Pode se observar que a interface possui links para as atividades, permitindo a edição das mesmas (Figura 6.19), e links para as ferramentas utilizadas para a execução das atividades (link "Ativar Microsoft Word").

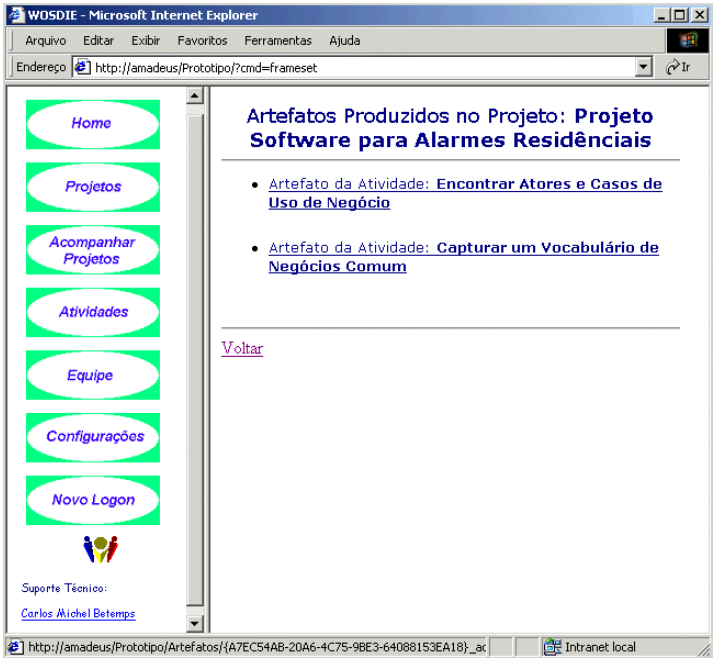

FIGURA 6.17 - Interface de Acesso aos Artefatos Concluídos de um Projeto

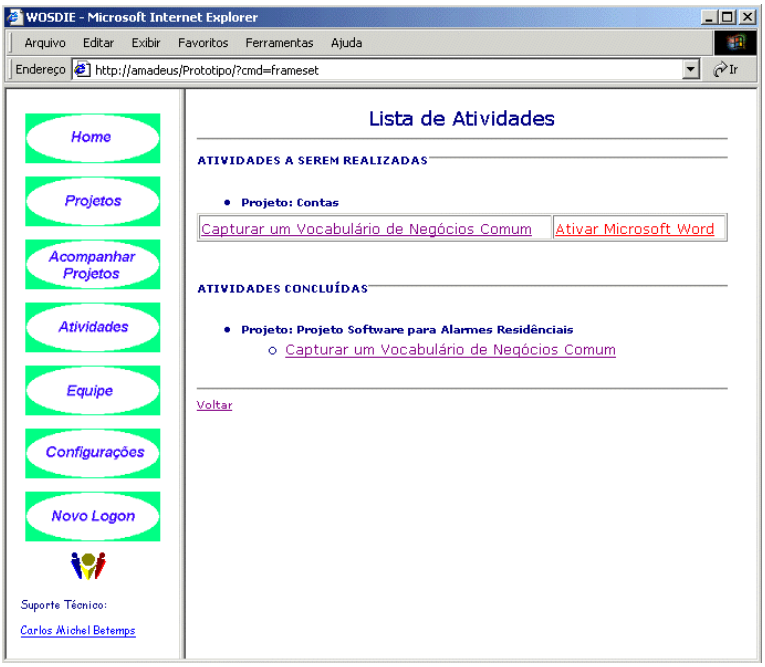

FIGURA 6.18 - Interface das Listas de Atividades

Quando o usuário clica nos *links* de ativação de ferramentas externas, o aplicativo correspondente é ativado no computador cliente. O mecanismo que permite esta integração entre ferramentas foi descrito na seção 6.4. Quando o *link* de edição das atividades é clicado, a interface da Figura 6.19 aparece. Através desta interface é possível editar os dados referentes à atividade como: percentual concluído, data de atualização e observações; e também pode se carregar para o servidor (*upload*), quando a atividade for concluída, os artefatos referentes a esta atividade.

Solicitação de Alteração: durante o processo de desenvolvimento de software, muitas vezes é necessário que sejam feitas modificações nos projetos ou mesmo são encontrados erros que necessitam correção. Para permitir que uma "Solicitação de Alteração" seja feita, o WOSDIE fornece interfaces especiais para isto. Quando uma solicitação de alteração for necessária, o usuário responsável por esta atividade (que pode ser qualquer usuário), quando for examinar sua lista de atividades, encontrará um *link* de atividade que dará acesso à interface da Figura 6.20. A partir desta interface o usuário poderá cadastrar uma nova “Solicitação de Alteração” ou editar alguma existente (Figura 6.21).

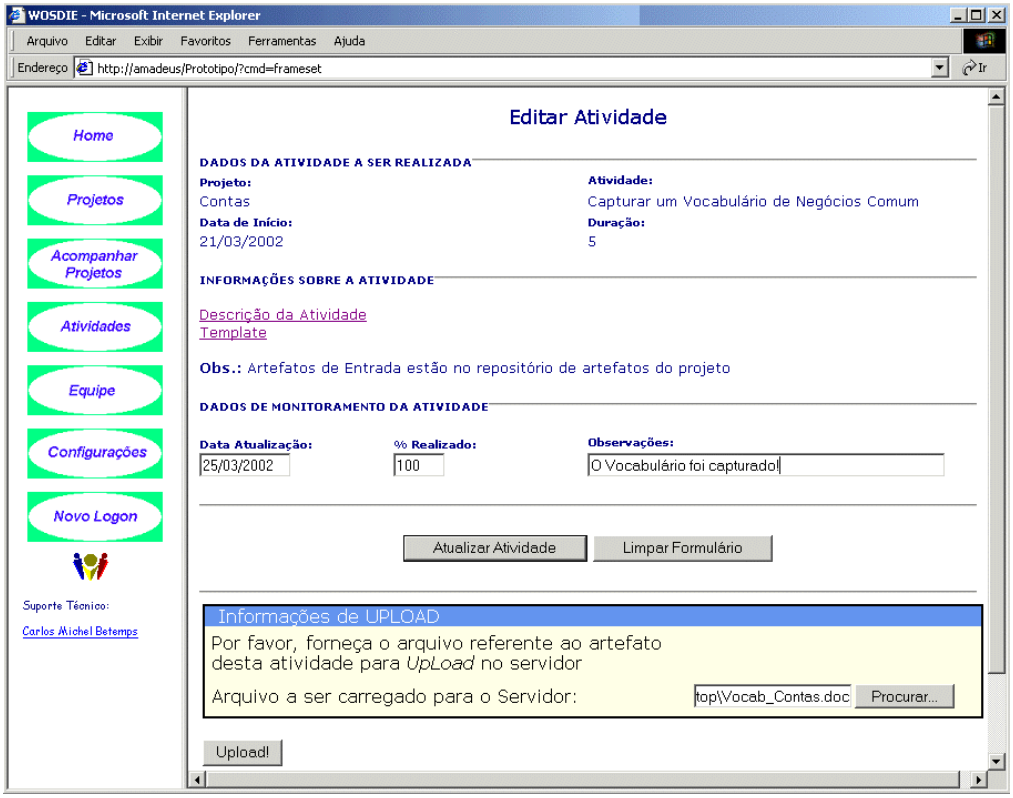

FIGURA 6.19 - Edição de Atividades e *Upload* de Artefatos

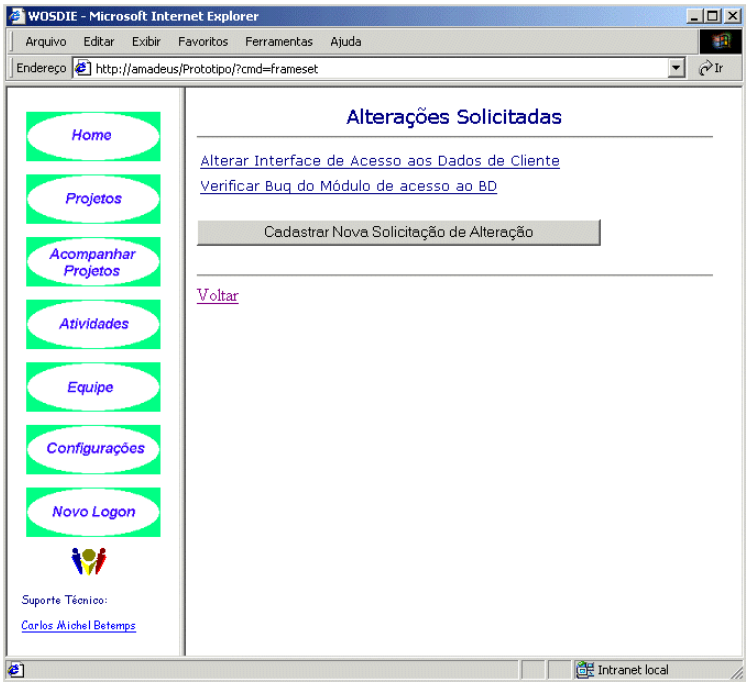

FIGURA 6.20 - Interface de Cadastro de Solicitações de Alteração

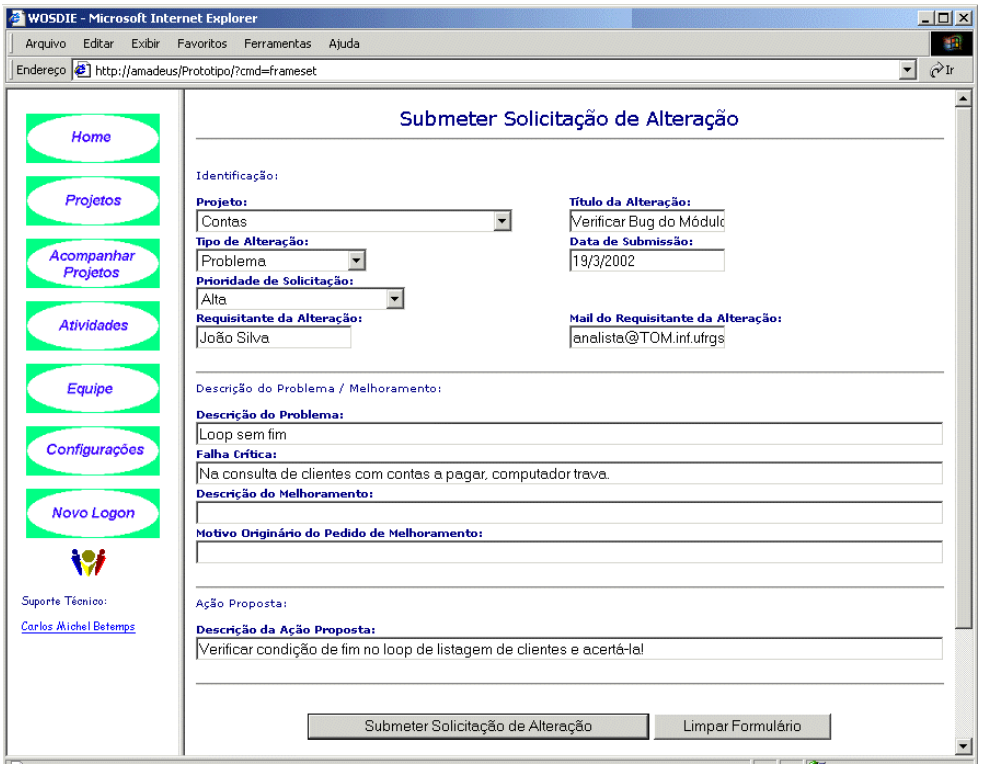

FIGURA 6.21 - Interface de Edição de Cadastro de Solicitações de Alteração

Equipe: a equipe de profissionais, que desempenham funções específicas dentro dos projetos, precisa ser cadastrada através de interfaces próprias. A Figura 6.22 mostra a interface que mostra os trabalhadores cadastrados e a Figura 6.23 mostra a interface que possibita a edição das informações relativas a um determinado usuário.

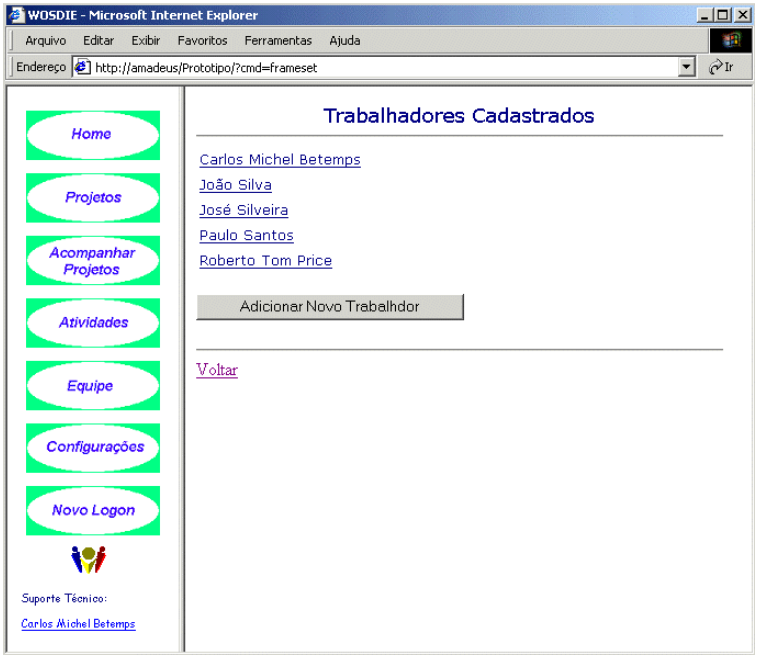

FIGURA 6.22 - Equipe Cadastrada no Ambiente

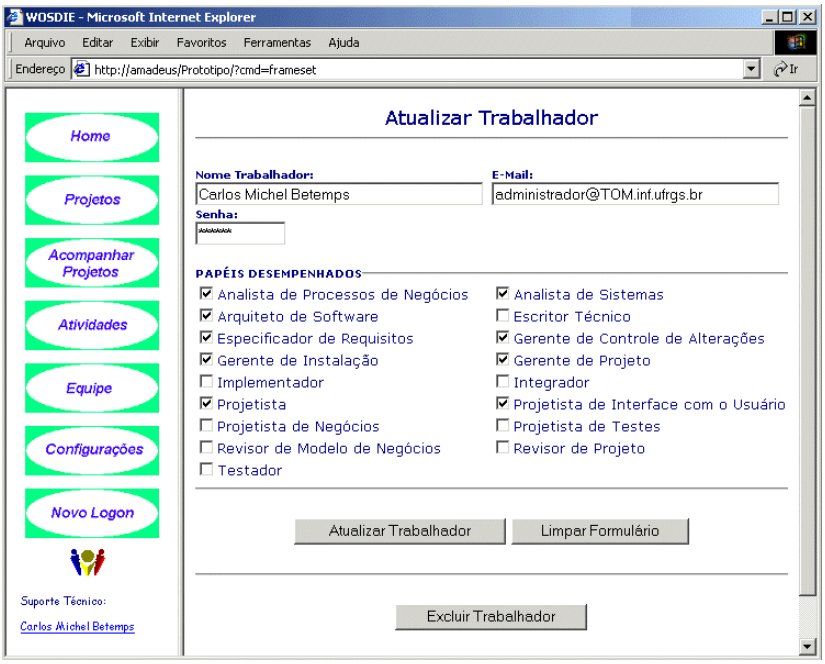

FIGURA 6.23 - Edição das Informações de um Usuário

Configurações: O *link* de "Configurações" ativa uma interface (Figura 6.24) que disponibiliza meios de configuração das atividades do processo de software seguido pelo ambiente (são as atividades do modelo de processo de software do protótipo e não as atividades referentes a um projeto em específico), das ferramentas integradas ao protótipo e dos papéis que cada participante da equipe pode desempenhar.

A Figura 6.25 mostra a interface de listagem dos papéis cadastrados no WOSDIE.

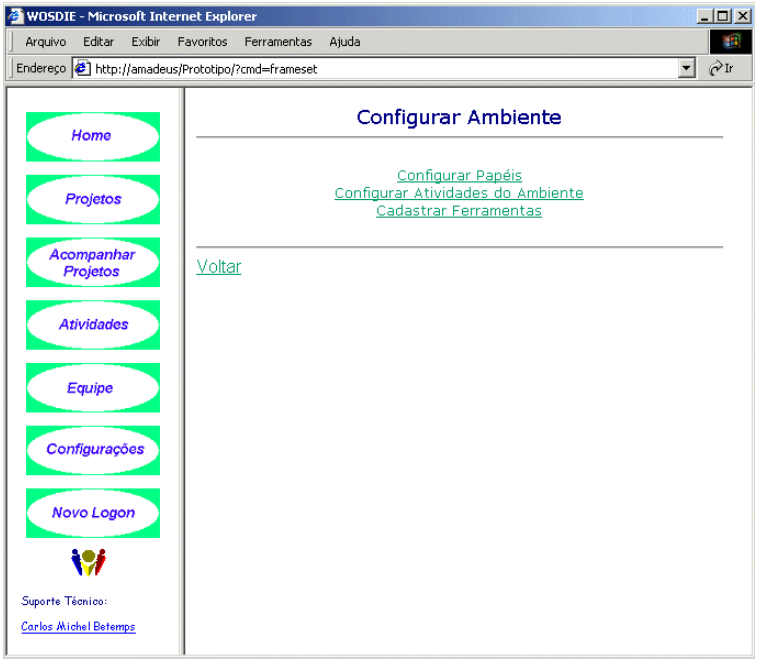

FIGURA 6.24 - Interface de Configurações do WOSDIE

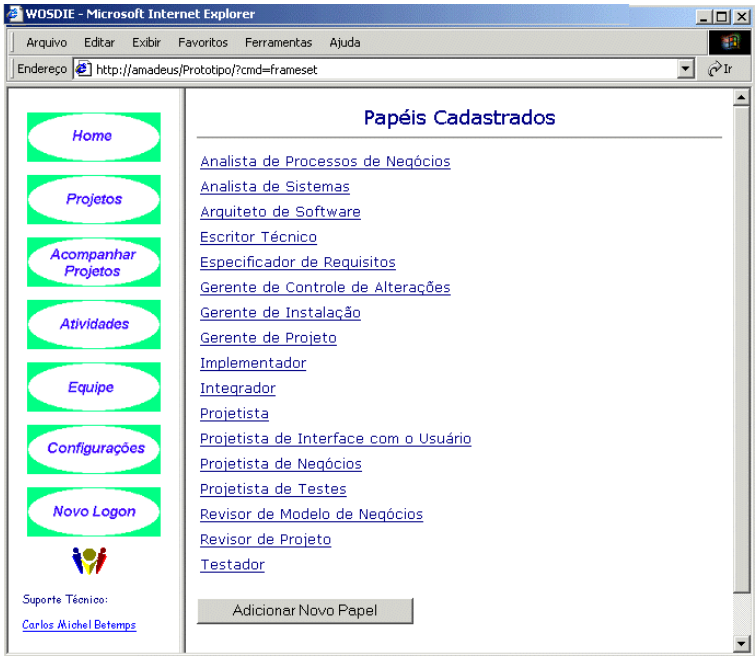

FIGURA 6.25 - Interface de Edição de Papéis (Funções)

A Figura 6.26 mostra a interface de listagem das atividades do ambiente, disponibilizando *links* que levam para a interface de edição das atividades do ambiente (Figura 6.27).

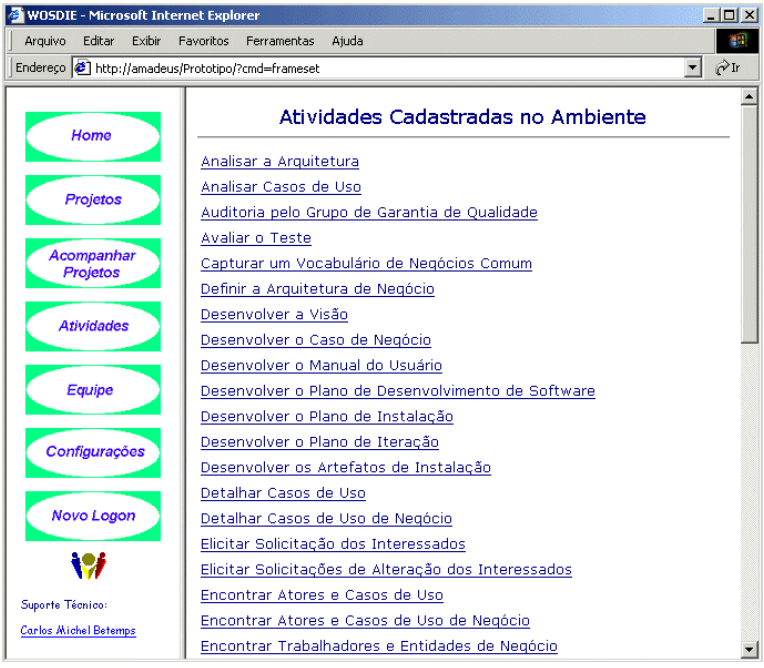

FIGURA 6.26 - Interface das Atividades do Ambiente

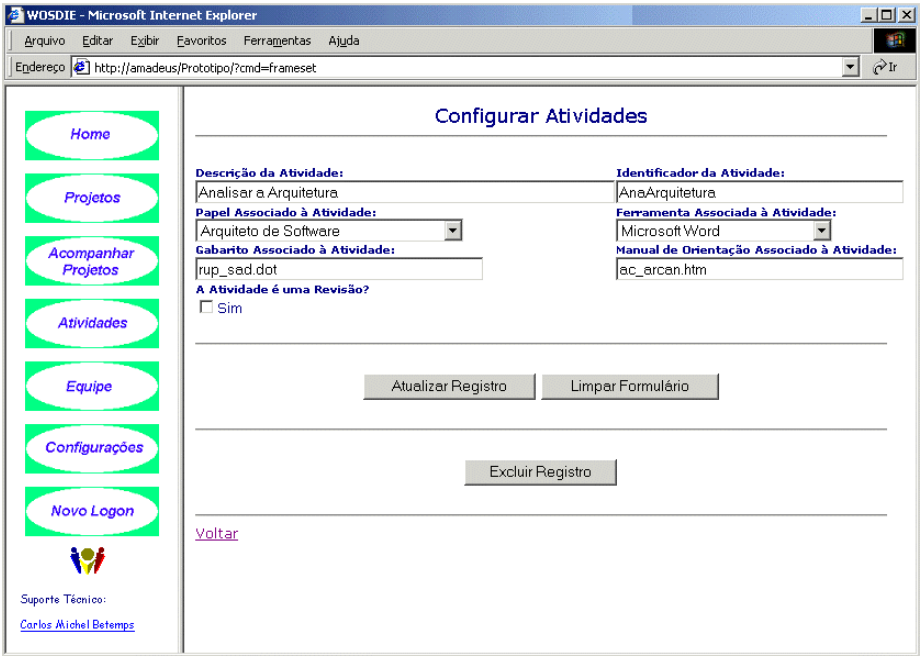

FIGURA 6.27 - Interface de Edição das Atividades do WOSDIE

A Figura 6.28 mostra os *links* de acesso as informações das ferramentas cadastradas no WOSDIE. Na Figura 6.29 é mostrado a interface de cadastro/edição das informações referentes as ferramentas de auxílio ao desenvolvimento de software cadastradas.

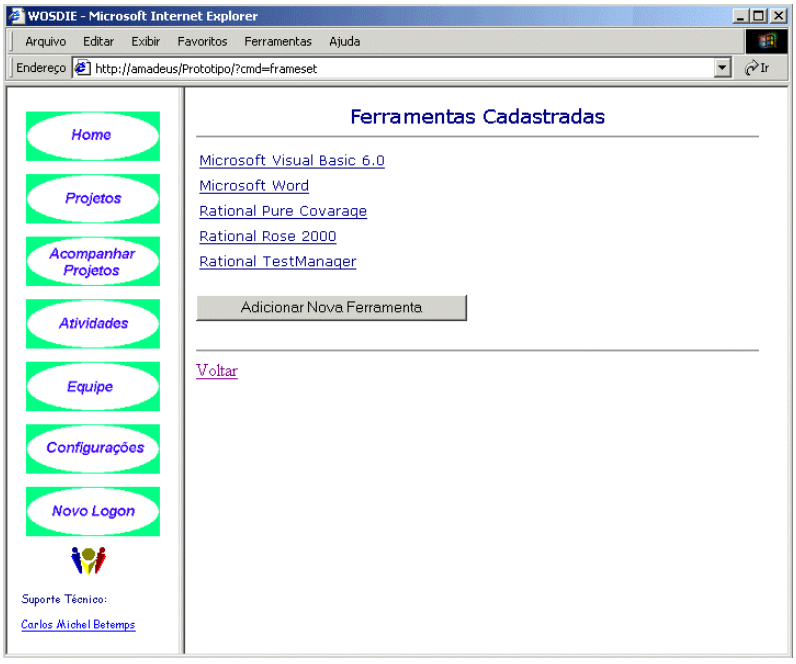

FIGURA 6.28 - Ferramentas Cadastradas no Ambiente

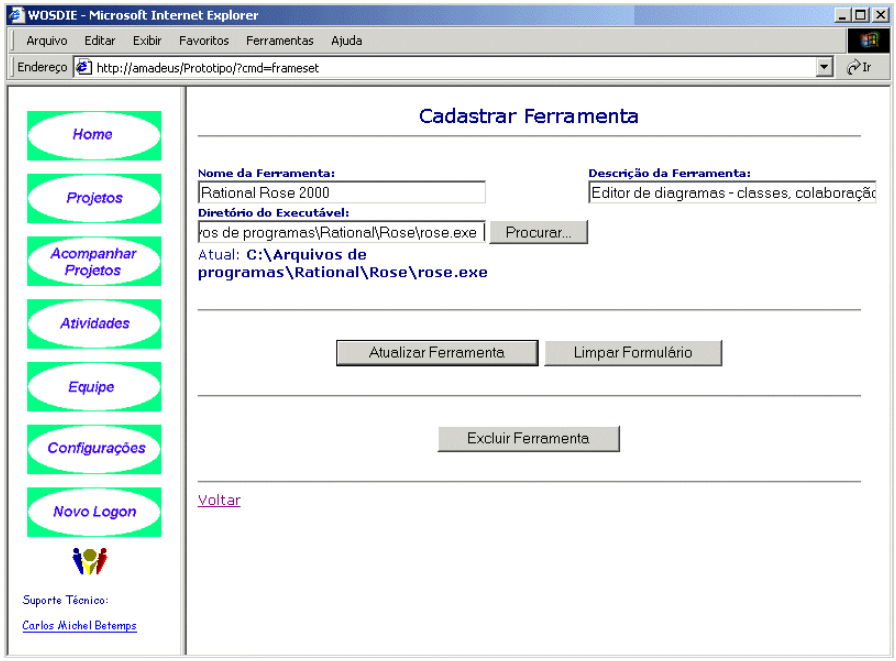

FIGURA 6.29 - Cadastro de Ferrametas do WOSDIE

Novo Logon: quando é necessário um novo *logon* no sistema, o usuário pode clicar no *link* "Novo Logon" e aparecerá a interface de *logon* no ambiente.

6.8 Casos de Uso do WOSDIE

Os usuários (trabalhadores, desenvolvedores, ou qualquer outro nome compatível) do WOSDIE são cadastrados no ambiente tendo papéis associados aos mesmos. Cada usuário pode realizar as atividades atuando diferentes papéis. Por exemplo, um trabalhador pode desempenhar, em uma atividade de teste, o papel de *Testador*, e, em uma atividade de geração de documentação técnica do produto, o papel de *Escritor Técnico*. Os papéis já cadastrados no WOSDIE são demonstrados em partes da Tabela 6.3 e da Tabela 6.4.

A Figura 6.30 mostra os casos de uso do WOSDIE. O papel de *Gerente de Projeto* é responsável pelos projetos de software do WOSDIE, isto é, cada novo projeto cadastrado no WOSDIE deve possuir um trabalhador (que é um *Gerente de Projeto*) responsável pelo mesmo; este trabalhador ficará encarregado da definição do responsável (atribuir o nome de um membro da equipe para a realização de uma atividade, de acordo com o papel exigido pela atividade), data de início e tempo estimado de cada atividade daquele projeto.

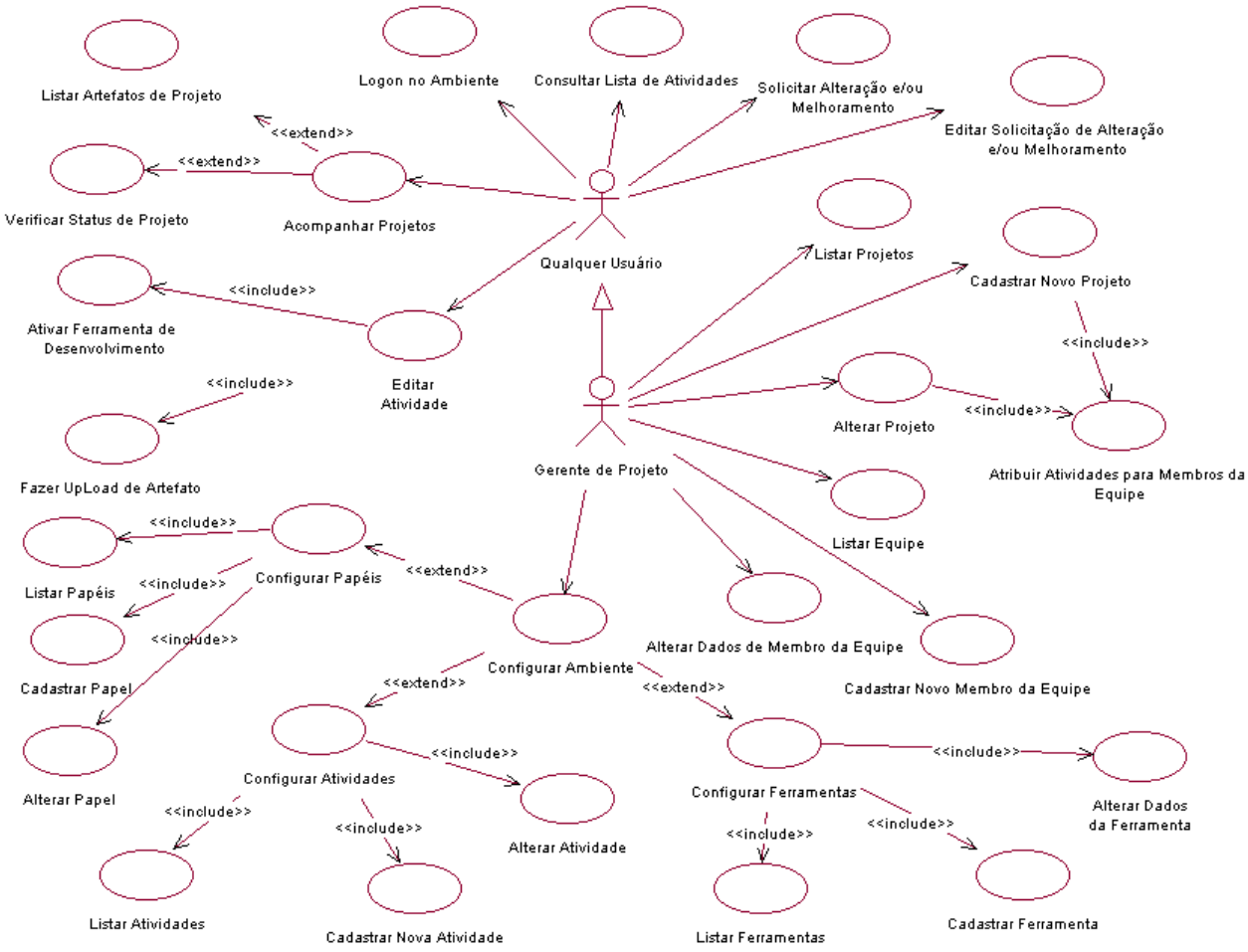

FIGURA 6.30 - Casos de Uso do WOSDIE

6.9 Avaliação do WOSDIE

Nesta seção será apresentada uma avaliação do WOSDIE. Esta avaliação será feita de acordo com uma *grade de avaliação* de ADS-CPs descrita no artigo "Assessing Process-Centered Software Engineering Environments" [AMV 97]. A grade de avaliação é dividida em três partes principais, cada uma dividida em alguns tópicos, que são [AMV 97]:

1. **Tecnologia PML (Linguagem de Modelagem de Processos)**: ADS-CPs existentes são baseados em uma variedade de linguagens de modelagem de processos. Para caracterizá-las são utilizados os seguintes tópicos:
    - **Escopo de Cobertura** - uma PML pode ser usada para melhorar a compreensão e documentação através de modelagem formal, análise e simulação. Isto pode ajudar no melhoramento do processo e iniciativas de reengenharia de processos. Um ADS-CP e sua PML podem oferecer suporte para uma ou mais fases do *ciclo de vida de processo* (ou *meta-*

processos), isto é, o conjunto de atividades que são seguidas para melhorar um processo de software.

- **Paradigma Lingüístico** - é o paradigma utilizado na linguagem. Por exemplo: orientado a objetos, baseado em regras, orientado a estados, etc.
- **Modelagem de Entidades do Processo** - entidades disponíveis para modelagem de processos (Atividades, Produtos, Ferramentas, Papeis, Organizações, etc.).
- **Modularidade/Composição/Reuso** - entidades de construção que suportem a modelagem de conceitos como modularização, composição e reuso.
- **Mecanismos para Execução e Evolução do Processo** - linguagens que utilizam interpretação, compilação e abordagens híbridas. Execução concorrente ou de um único processo de execução. Mecanismos de evolução como características reflexivas.
- **Controle de Cooperação e Concorrência** - construções lingüísticas para a modelagem de cooperação (entre humanos) e controle de concorrência - podem ser distinguidas dois tipos de cooperação: síncrona (teleconferência, *chats*) e assíncrona (e-mail)
- **Suporte Metodológico** - suporte metodológico para as diferentes fases do ciclo de vida de processo.
- **Suporte de Ferramentas da PML** - editores, simuladores, interpretadores, processadores de texto, etc.

2. **Arquitetura do ADS-CP**: a arquitetura de ambientes deste tipo podem ser caracterizadas pelos seguintes aspectos.
   - **Arquitetura de Alto Nível** - caracterização dos ADS-CPs de acordo com os seus principais componentes e a integração e interação entre os mesmos.
   - **Interação com ferramentas de produção e facilidades de integração de ferramentas** - mecanismos básicos para integrar ferramentas de produção externas ao ambiente.
   - **Comportamento do ADS-CP** - comportamento do ambiente do ponto de vista do usuário, isto é, o paradigma de interação com o usuário do ADS-CP do ponto de vista dos desenvolvedores e por outros agentes do processo, com gerentes de projeto.
   - **Integração de Dados** - mecanismos para a integração de dados e estruturas/papéis do repositório dentro da arquitetura do ADS-CP.
   - **Gerência de Inconsistência e Sincronização de *Workspace*[‡]** - gerência do ambiente de trabalho onde ferramentas acessam seus dados, gerência de (possíveis) inconsistências entre os *workspaces* e o(s) banco(s) de

[‡] Workspace - se refere ao ambiente personalizado de cada usuário.

dados do ADS-CP e identificação de operações acompanhadas fora do controle do ADS-CP.

- **Distribuição e Heterogeneidade** - distribuição do ambiente (centralizado, cliente-servidor, *peer-to-peer*), e arquiteturas heterogêneas versus arquiteturas homogêneas.
- **Suporte para Métricas** - suporte para a coleção e avaliação de processos e métricas de produto.
- **Eficiência da Implementação** - tópicos de performance e adoção/integração de protótipos para suporte ao desenvolvimento de ADS-CP.

3. **Experiências**: observações podem ser derivadas de experimentos práticos com o ADS-CP, classificados como:
    - **Exemplos de Referência** - aplicações do ADS-CP e sua PML para exemplos de referência (*ISPW-6* e *ISPW-7*, *RUP*, *OPEN* e *OOSP*)
    - **Experimentos Internos** - aplicações internas do ADS-CP/PML para exemplos e casos desenvolvidos dentro do grupo de desenvolvimento ou derivados de projetos industriais.
    - **Projetos Industriais** - aplicações do ADS-CP/PML em projetos industriais.

A seguir cada aspecto definido na grade de avaliação acima será avaliado em relação ao WOSDIE.

6.9.1 Tecnologia PML

Escopo de Cobertura

O WOSDIE permite a modelagem e execução de todas as fases do ciclo de vida de software - desde a elicitação de requisitos e modelagem de negócios até a implementação, testes e implantação. A PML do WOSDIE é a linguagem do *Workflow Designer*, que é a ferramenta de modelagem de processos de workflow do *Exchange 2000 Server*. A ferramenta *Workflow Designer* fornece uma linguagem de modelagem gráfica, que possui entidades de "estados", "ações" e "transições entre estados" (esta última é um tipo especial de "ação").

Avaliação:

1 - A linguagem de modelagem do ambiente proporcina a modelagem de toda a extensão do modelo de ciclo de vida de software. Foi modelado no ambiente WOSDIE um modelo de processo de software extraído do processo unificado da *Rational* (RUP). Este processo abrange todos os *workflows* definidos no processo RUP. Os modelos dos workflow disponibilizados pelo RUP estão na notação de diagramas de atividades. Esta notação teve que ser "transformada" para a notação da ferramenta

Workflow Designer. Esta transformação implica na criação de estados, ações e transições entre estados para cada atividade modelada nos modelos do RUP.

2 - A linguagem de modelagem é orientada a "estados". Possui também os conceitos de "ação" e de "transicão entre estados".

Paradigma Linguístico

A linguagem de modelagem do protótipo de experimentação é baseada em "Estados", "Ações" e "Transições". Possui os campos "Expressão de Condição" e "Procedimento de Ação" para cada "Ação" definida no modelo do processo. Os modelos de processos modelados na linguagem do WOSDIE podem ser classificados como "Modelos de Transição de Estados" (Paradigma de "Modelos de Transição de Estados").

O modelo de processo define os estados em que um determinado item pode estar - um item de projeto. Um item de projeto possui propriedades que são avaliadas no decorrer da execução do processo (estas propriedades são avaliadas nos campos "Expressão de Condição"), e, de acordo com estas propriedades, os itens vão causando a execução de procedimentos de ação e transições entre estados (que também possuem procedimentos associados).

A cada nova "Expressão de Condição" que alcança o valor "Verdadeiro" na execução de processo, uma ação correspondente pode ser executada, podendo-se:

- Criar um novo item de atividade, assinalando uma atividade a uma pessoa;
- Executar uma atualização nos itens de dados do ambiente;
- Enviar mensagens de mail eletrônico aos participantes do grupo de desenvolvimento;
- etc..

Modelagem de Entidades do Processo

As entidades disponíveis na linguagem de modelagem são as seguintes:

- **Estados** - um item de projeto pode passar de estado para estado dentro de um modelo de processo.
- **Ações** (Transições) - Ações e transições são procedimentos definidos para que ocorram em determinadas condições (Expressões de Condição). A diferença entre estas é que uma Ação não causa uma troca de estados e uma Transição faz com que um item passe de um estado para outro. Os conceitos de Ação e Transição são praticamente o mesmo, a única diferença é que o estado origem em uma Ação é o mesmo estado destino de um item sendo "guiado" por um modelo de processo.
- **Itens do *Exchange*** - existem itens que podem ser criados a partir da biblioteca CDO [MIC 2001a] e que podem ser utilizados pelo motor de *workflow* do *Exchange*. Estes itens permitem que entidades como atividades, papéis, trabalhadores (atores), ferramentas, solicitações de alteração, etc. sejam criadas e façam parte do processo modelado. A partir de um modelo de processo (um modelo contendo estados, ações e transições) definido para controlar os estados em que um item de projeto pode estar, o WOSDIE pode desencadear a criação de itens de atividades, equipes de revisão, atualização de itens de projeto, assinalamento de atividades, etc. Deste modo, um item de projeto é uma composição de vários outros itens (atividades, equipe, ferramentas, etc.), definindo e caracterizando cada um dos

projetos de software em andamento ou já concluídos dentro do ambiente WOSDIE, e é através das propriedades (seu estado - *status*) de um item de projeto (e os itens que o compõem) que o motor de *workflow* do *Exchange* gere a atribuição e distribuição de atividades dentro de um projeto de software. Através das interfaces do protótipo podem ser definidos Trabalhadores (Equipe), Papéis, Ferramentas, Atividades (Modelos de Atividades definindo o papel, gabarito, manual de orientação, ferramenta, etc. associados a atividade), etc..

Modularidade/Composição/Reuso

O protótipo WOSDIE usa o modelo de processo definido no *Workflow Designer* do *Exchange*. Este modelo de processo pode ser reutilizado a cada novo projeto que é cadastrado no WOSDIE, gerenciando as atividades associadas ao projeto. Também podem ser reutilizados gabaritos, manuais de atividades e todo o conjunto de documentos associados ao processo RUP. Artefatos já construídos em projetos anteriores podem ser analisados pelos desenvolvedores em uma fase posterior no mesmo projeto e em outros projetos, podendo fazer reuso destes artefatos quando necessário.

A definição do modelo de processo realizado na ferramenta *Workflow Designer* é feito como um modelo completo, isto é, o modelo de processo não é dividido em fases ou pacotes. As atividades relacionadas a cada workflow definido pelo RUP são modeladas em um mesmo modelo. Como pode ser visualizado na Figura 6.4, o modelo do processo abrange todos os *workflows* do processo unificado da *Rational* (RUP).

Avaliação das Características Básicas de uma PML

- **Integração de Ferramentas**: a integração de ferramentas no WOSDIE não é definida pela linguagem de modelagem. No modelo de processo, quando uma nova atividade é assinalda para um usuário (um participante da equipe), um novo item de "Atividade" é criado. Nas interfaces de acesso do protótipo (programadas utilizando ASP e *JavaScript* e o dispositivo de cadastro de formulários do *Exchange* - Seção 6.1 - que permite que itens sejam visualizados a partir de documentos ASP apropriados), de acordo com a atividade sendo editada, são criados links que ativam a ferramenta associada a mesma. A definição de modelos de atividades, ferramentas a serem utilizadas, papéis que podem ser desempenhados, etc. é realizada também a partir de interfaces do WOSDIE.

Os *links* de ativação de ferramentas utilizam funções *JavaScript*, que por sua vez utilizam o servidor de automação do *windows* ou componentes *ActiveX* que permitem que ferramentas externas sejam ativadas a partir de um navegador *Web* (Seção 6.4).

- **Suporte para a Evolução**: o modelo de processo do WOSDIE pode ser modificado, mesmo que projetos estejam em execução. A modificação dos processos necessita de dois tipos de alteração:

1) Na ferramenta *Workflow Designer* é necessário que os estados, ações e condições associadas as modificações do processo sejam modeladas;
2) Nas interfaces de configuração de atividades do WOSDIE, é necessário que as configurações de atividades sejam modificadas e/ou que as novas atividades que estão sendo adicionadas ao modelo sejam configuradas.

- **Modelagem de Atividades Concorrentes**: a modelagem a nível conceitual de atividades concorrentes não é possível na *Workflow Designer* (como descrito na seção 3.6.6.4). Este tipo de modelagem pode ser simulada pela instanciação de múltiplos itens de atividades em uma mesma ação (transição de estado).

- **Modelagem de produtos**: cada atividade tem os seus produtos associados, e estes produtos são tratados como propriedades das atividades, como: gabaritos, manuais, ferramentas, papel (função - *role*), etc.. Estas propriedades são configuradas através das interfaces do WOSDIE. Assim, cada atividade pode ser definida e configurada de maneira apropriada.

- **Mecanismos de Integração**: dentre as ferramentas ativadas pelo WOSDIE, dependendo se a atividade em execução possui algum gabarito associado, o WOSDIE ativa a ferramenta e já abre este gabarito para a edição pelo desenvolvedor. Por exemplo: a ativação do *Microsoft Word* e a abertura dos gabaritos (*templates*) fornecidos pelo RUP e a ativação de ferramentas do *suite Rational* [RAT 2001]. Outro tipo de integração é a geração dinâmica de um documento que possui *links* para os artefatos produzidos nos projetos (integração de dados). A integração de dados do WOSDIE necessita ser trabalhada. As ferramentas de desenvolvimento precisam salvar seus documentos em formatos de arquivos compatíveis, visto que não existe um mecanismo integrador de dados no WOSDIE que seja capaz de armazenar os dados em um formato qualquer oriundo de uma ferramenta e permitir que ferramentas diferentes utilizem estes dados, a não ser se estão em um mesmo formato. Duas maneiras de realizar a completa integração de dados foram apresentados na seção 2.1, que são: dicionários de dados [NOT 2000] [OIN 99] [JAR 94] e documentos XML [XML 2001].

- **Facilidades de Modularização**: o modelo de processo do WOSDIE não é dividido em fragmentos. O desenvolvimento, a execução e a manutenção do modelo de processo devem ser feitos sobre o processo completo.

- **Mecanismos básicos para suportar a gerência de Workspace** (Ambiente de Trabalho de cada desenvolvedor): de acordo com o desenvolvedor que se loga no protótipo, uma lista de atividades diferentes aparece. O banco de dados do WOSDIE é consultado sobre as atividades assinaladas ao usuário logado. Outra consulta deste tipo é feita em relação às ferramentas cadastradas no protótipo. De acordo com o endereço IP da máquina cliente que está sendo utilizada por um desenvolvedor, diferentes ferramentas de desenvolvimento podem estar cadastradas.

- **Mecanismos básicos para descrição de controle de concorrência**: concorrência é obtida através da criação de vários itens de atividades ao mesmo tempo. Não são criadas atividades que possuam mais de um trabalhador assinalado para a mesma. Concorrência só pode existir entre atividades, e não intra-atividades.

- **Características Reflexivas**: não existe nenhuma característica reflexiva no protótipo WOSDIE. Características reflexivas, mais especificamente objetos reflexivos, para o controle, gerência e monitoramento de outros objetos em ambientes de desenvolvimento de software é proposto por Yamaguti [AMA 2002] em sua tese de doutorado.

Mecanismos para Execução e Evolução do Processo

A execução do modelo de processo do protótipo WOSDIE é realizada pelo SGW *Exchange 2000 Server* (E2K) (pelo motor de *workflow* do E2K). A cada criação

de um novo projeto de software através das interfaces do WOSDIE, uma instância do modelo de processo é criada. Assim, cada projeto é "conduzido" separadamente pelo WOSDIE e obedecendo ao mesmo modelo de processo. O modelo de processo é armazenado em uma das pastas do *Web Storage System* (WSS) - a pasta "Projetos". A execução dos processos no WOSDIE é feita de maneira interpretada, isto é, o E2K interpreta as condições e dispara ações e troca de estados a partir da interpretação do modelo de processo e da análise das condições de execução.

Controle de Cooperação e Concorrência

Cooperação no WOSDIE é obtida através da possibilidade de se ter vários usuários logados no ambientes ao mesmo tempo, realizando atividades como verificação de listas de tarefas (atividades), edição de atividades (revisões), *upload* de artefatos (documentos, diagramas, códigos-fonte, etc.), etc.. Cada usuário pode verificar suas próprias atividades (as atividades assinaladas para o mesmo), mas não pode verificar atividades assinaladas a outros usuários. A cooperação ocorre entre as atividades de um projeto e não dentro de uma mesma atividade (cooperação no nível de projeto).

Concorrência ocorre ao nível de projetos, por exemplo: temos dois usuários acessando o ambiente e editando suas atividades, mas cada atividade é pertencente a um projeto diferente. Cada projeto é conduzido de maneira independente, podendo assim haver a concorrência entre as atividades de projeto diferentes. Concorrência em um mesmo projeto, ou mesmo atividades paralelas, são possíveis, mas é necessário que, ao nível de modelagem dos modelos de processo na ferramenta *Workflow Designer*, sejam criadas mais de uma atividade em uma ação de transição entre estados; esta transição assinala as atividades aos participantes da equipe de desenvolvimento.

Cooperação na edição de um artefato (por exemplo: modelo de classes) pode ser realizado, basta que os desenvolvedores troquem entre si o documento em edição, cada um fazendo suas alterações e propostas. Uma observação importante em relação ao WOSDIE é que uma atividade sempre tem um único responsável, isto é, vários desenvolvedores podem trabalhar em uma atividade na edição de algum artefato, mas a atividade so terá um responsável, e, portanto, para o WOSDIE somente um desenvolvedor criou o artefato resultante da atividade. A integração de uma ferramenta de edição colaborativa, como a proposta por Cayres [CAY 99] e outros, poderia resolver esta questão de cooperação ao nível de um único artefato.

Suporte Metodológico

Não existe nenhum suporte metodológico que guie o desenvolvimento de um modelo de processo e sua execução, extensão e avaliação dentro do WOSDIE. O modelo de processo modelado na ferramenta *Workflow Designer* foi extraído do processo RUP. RUP define vários *workflows* (processos) que definem as atividades a serem desenvolvidas durante uma iniciativa de desenvolvimento de software. Os passos e mecanismos de modelagem utilizados para a definição de um modelo de processo na ferramenta *Workflow Designer* foram descritos na seção 6.3.1. O processo de modelagem foi seguido de acordo com o modelo de processo descrito através do diagrama de atividades da Figura 6.4 e aplicado de acordo com a seção 6.3.1.

Suporte de Ferramentas da PML

Toda modelagem de processo é feita graficamente na ferramenta *Workflow Designer*. Os *scrips* de ações e condições são programados em *VBScript*. O WOSDIE fornece, através de sua interface de acompanhamento de projetos, informação sobre em

que estado se encontra um projeto. A integração de ferramentas é realizada através da criação de *links* a partir do cadastro de ferramentas no WOSDIE por parte dos usuários. Esta integração é realizada pela geração de *links* para funções *JavaScript* que utilizam controles *ActiveX* e o Servidor de Automação do *Windows* para ativar as ferramentas.

6.9.2 Arquitetura do ADS-CP

Arquitetura de Alto Nível

WOSDIE é baseado na plataforma *Internet*. Funciona em um servidor *WEB* que possui o sistema operacional *Windows 2000 Server* e, por conseqüência, o Exchange 2000 Server (E2K). Algumas ferramentas de desenvolvimento são ativadas diretamente a partir do WOSDIE. O *Web Storage System* atua como o repositório do WOSDIE. O E2K, mais especificamente o motor de *workflow* do E2K, atua como o motor de processo do WOSDIE. Os modelos de processo são definidos na ferramenta *Workflow Designer*. A ativação de ferramentas é feita de dentro do *navegador Web*, utilizando componentes e objetos *ActiveX*.

A Figura 6.10 descreve os componentes da arquitetura do WOSDIE.

Interação com ferramentas de produção e facilidades de integração de ferramentas

O WOSDIE é executado sobre a plataforma internet, rodando em um navegador *Web*. A interação com ferramentas de produção como *Microsoft Word, Rational Rose*, etc., é realizada via navegador através de dispositivos de *Automação OLE* do *Windows*. Os meios de ativação de ferramentas externas ao WOSDIE são descritos na seção 6.4 (Ativação de Ferramentas a Partir do WOSDIE).

As ferramentas que são utilizadas no WOSDIE devem ser cadastradas para que possam ser associadas com atividades específicas do processo de desenvolvimento, por exemplo, associar o processador de textos *Microsoft Word* (ou qualquer outro) a atividade de "Desenvolver o Manual do Usuário".

Comportamento do ADS-CP

A interface do WOSDIE foi construída a partir de documentos ASP e possui a aparência e funcionalidade como a de qualquer página *Web*. O acesso a dados e a configuração e ativação de ferramentas é feito utilizando-se *links* e botões, como os normalmente encontrados nas páginas *Web*. Existe um quadro (*frame*) que serve como um menu principal e, a partir do mesmo, pode-se acessar todas as funcionalidades do WOSDIE. As interfaces de acesso do WOSDIE podem ser visualizadas na seção 6.7 (Interfaces de Acesso).

Integração de Dados

WOSDIE utiliza o WSS (*Web Storage System*) do E2K, o qual é um banco de dados semi-estruturado que torna possível o armazenamento de documentos dos mais diversos tipos. O WSS é utilizado para armazenar os artefatos produzidos ao longo processo de desenvolvimento de software. O WSS mantém os artefatos como em um sistema de arquivos comum, isto é, mantém os artefatos, em forma de arquivos, em pastas definidas para este propósito.

Os artefatos, que podem ser arquivos dos tipos mais variados, produzidos pelos participantes do processo podem ser carregados (*upload*) para o servidor que mantém o WOSDIE. Estes artefatos podem ser acessados posteriormente através de *links* criados dinamicamente (através de documentos ASP que geram estes *links* de acordo com os artefatos, já produzidos, associados a cada projeto) pelo WOSDIE (estas funcionalidades podem ser visualizadas na Figura 6.17 e na Figura 6.19).

A integração de dados do WOSDIE pode ser melhorada através da criação de documentos XML que podem manter as informações relacionadas as atividades como: responsável, artefatos associados, projeto, percentual realizado, gabaritos associados, etc..

Outra forma de integração e compartilhamento de dados seria a utilização de um banco de dados comum, necessitando para isso, uma ferramenta que forneça meios de armazenar e acessar dados vindos de diferentes ferramentas (como dicionário de dados).

Gerência de Inconsistência e Sincronização de Workspace

O acesso ao WOSDIE é feito através de um navegador *Web* e de um endereço internet (http://amadeus/Prototipo ou http://143.54.8.164/Prototipo). Cada usuário do WOSDIE possui uma senha de acesso ao ambiente. Quando o *login* é feito com sucesso, são disponibilizados ao usuário todos os links que podem ser acessados pelo mesmo. As informações de atividades referentes a determinado usuário são disponibilizadas para este usuário, e somente para este, através do *link* "Atividades" (no menu principal) do WOSDIE. Este *link* pode ser visualizado na maioria das figuras da seção 6.7 (Interfaces de Acesso do WOSDIE). Os artefatos produzidos pelos participantes do processo, e que já foram carregados para o servidor, não podem ser apagados através das interfaces do WOSDIE. Isto só é possível se feito no próprio servidor do WOSDIE.

Distribuição e Heterogeneidade

WOSDIE foi implementado através de um conjunto de documentos ASP (que formam a interface do WOSDIE e recuperam as informações a respeito dos projetos de software), de um SGW (o *Exchange 2000 Server*, e que faz a gerência dos processos do ambiente) e de um banco de dados semi-estruturado (o WSS - que armazena as informações e documentos associados aos projetos).

O WOSDIE é acessado via navegador *Web* e pode ser acessado de qualquer lugar ao redor do planeta, necessitando apenas que o usuário possui acesso a internet. Esta característica possibilita a distribuição de tarefas entre vários profissionais.

Uma restrição em relação a heterogeneidade do WOSDIE surge devido a utilização do *Exchange 2000 Server* (e por conseqüência o *Windows 2000 Server*). Durante a construção do WOSDIE, mais especificamente durante os testes realizados com o WOSDIE, foi notado que somente computadores que possuíam o *Microsoft Internet Explorer* como navegador e o *Microsoft Windows NT* ou *WINDOWS 2000 Server* como sistema operacional consiguiram acessar o WOSDIE da maneira desejada, os demais sistemas operacionais e navegadores conseguem acessar o endereço internet relativo ao WOSDIE, mas não acessam as mesmas interfaces nem fazem com que o motor de workflow do Exchange seja ativado.

Suporte para Métricas

WOSDIE não possui nenhum mecanismo de suporte para métricas. A coleção de métricas seria uma funcionalidade interessante para qualquer ambiente de desenvolvimento de software. Medidas como: tempo de edição de artefatos, percentual de atividades finalizadas nos projetos, tempo de atrazo na finalização de projetos, etc. são interessantes no sentido que permitem o monitoramento dos projetos, e este, é de crucial importância na atualidade. Coleção de métricas e adesão de um processo de desenvolvimento disciplinado são tarefas difíceis em qualquer projeto de software [CAL 96] e métricas não podem ser definidas totalmente para todas organizações, é necessário que as mesmas sejam selecionadas de acordo com as metas, características e problemas de cada organização em específico [AMV 97].

Eficiência da Implementação

WOSDIE demonstrou alguns problemas de eficiência. A gravação, leitura e edição de informações (e ou arquivos) dentro do WSS é lenta. O monitoramento dos eventos dentro do ambiente por parte do SGW do Exchange demonstrou-se ser eficiente, mas quando um evento causa a troca de estados dentro do modelo de *workflow* do WOSDIE (ver Seção 6.3.1), a interpretação dos *scripts* associados as trocas de estado, e a gravação das informações associadas, tornam este tipo de operação também lenta.

A disponibilização das interfaces (geração de *links* apropriados) do WOSDIE é feita através da execução dinâmica de *scripts* existentes dentro de documentos ASP. Estes *scripts* acessam informações do WSS e fazem consultas SQL ao banco de dados do WOSDIE (WSS). Este tipo de operação normalmente tem um tempo de resposta lento, mas aceitável, visto que se está acessando dados via *Web*.

Foram feitos testes onde o servidor que mantém o WOSDIE foi acessado por dois usuários diferentes, a partir de máquinas também diferentes, e foi constatado que a eficiência tem um decréscimo bem alto, visto que o WSS tem que lidar com a sincronização e controle das transações que partem destas máquinas.

O *Exchange 2000 Server* (E2K) atuando como Sistema de Gerência de *Workflow* e, por conseqüência, gerindo as atividades do processo de software, causa a baixa eficiência do WOSDIE nas situações descritas anteriormente. O modelo de implementação do E2K, utilizando uma notação parecida com diagrama de estados, e a utilização de programação de *scripts* (*VBScript*) como dispositivo para a realização de gravação, leitura e atualização de informações do banco de dados faz com que o WOSDIE tenha um tempo de resposta lento em determinadas situações.

6.9.3 Experiências

Exemplos de Referência

O processo modelado no WOSDIE (através do *Workflow Designer*), descrito na seção 6.3, foi extraído do RUP (processo unificado Rational). Este processo, proposto pela Rational Corporation, têm sido cada vez mais implantado dentro das organizações de desenvolvimento de software [RAT 2001]. O WOSDIE também utiliza documentação e gabaritos (*templates*) fornecidos pelo RUP.

Experimentos Internos

O WOSDIE foi testado por alguns componentes do Grupo Amadeus [AMA 2002]. Nestes testes alguns projetos simplificados foram configurados no WOSDIE e executados. Estes testes proporcionaram, principalmente, a observação das atribuições de atividades aos responsáveis por parte do SGW (E2K) de acordo com o modelo de processo.

Projetos Industriais

O WOSDIE não foi experimentado em nenhum projeto industrial.

Na realidade, a experimentação de ambientes de modelagem e execução de processos de software, como o WOSDIE, em ambientes de produção de software reais é uma tarefa muito difícil, já que, como defendido por Ambriola et. al. [AMV 97]:

- As organizações de produção de software normalmente não possuem uma preocupação com o processo de software em utilização, ou mesmo não possuem um processo explicitamente definido.
- Outro fator importante é que as organizações procuram mais é obter ferramentas de auxílio a execução dos processos, enquanto o mais crítico ocorre em relação a compreensão, análise e documentação dos processos.
- A introdução de uma nova tecnologia é muito complicada, pois geralmente afeta fortemente as atividades em andamemnto e necessita de muito tempo e investimento para que possa ser utilizada realmente.
- E ainda há outro fator, é necessário um melhor entendimento de como avaliar o impacto que um software de análise, modelagem e execução de processos pode ter sobre um ambiente de produção realístico.

6.10 Possíveis Extensões para o WOSDIE

O WOSDIE é um ambiente que pode ser extendido em vários aspectos. Dentre estes podemos citar os seguintes:

- **Integração com Ferramentas de Gerência de Projetos**: a integração do WOSDIE com ferramentas de gerência de projetos (como o *Microsoft Project* [MIC 2001]) é uma possibilidade interessante, e talvez fundamental, visto que, ferramentas deste tipo permitem o planejamento das atividades, geração de relatórios sobre atividades atrasadas e visualização do impacto destas sobre o projeto como um todo, e várias outras funcionalidades.

- **Coleta de Métricas:** a coleta de dados a respeito da execução das atividades do processo pode ser útil para a extensão e melhoramento do próprio processo, tanto ao nível de execução e modelagem, e também do ambiente. Dados como tempo de edição dos artefatos gerados durante o processo, duração de atividades (ou revisões), resultados positivos e/ou negativos relativos às etapas de implementação (baseados nos resultados de testes), etc., podem ser coletados ao longo da execução dos projetos e armazenados através de algum dispositivo (por exemplo: itens do *Exchange*). Estas dados podem ser utilizados posteriormente para a geração de dados estatísticos e

conseqüente medidas em relação a execução dos projetos e do próprio WOSDIE, rendimento dos participantes do processo, etc.

- **Melhorar os Níves de Segurança de Acesso ao Ambiente**: o WOSDIE possui um mecanismo muito simples de acesso. Uma possível extensão do WOSDIE poderia ser a melhora dos dispositivos de acesso e identificação. Uma alternativa seria a modificação do WOSDIE para rodar em um ambiente *internet* de "**https**" (http seguro), como o utilizado em muitos provedores de correio eletrônico baseados na *internet* (os conhecidos sistemas *WebMail*).

- **Utilização de Componentes mais Portáveis e Software Livre**: Implementação do WOSDIE com a utilização de PHP e MySQL, para oferecer maior portabilidade, e componentes mais portáveis, que permitam a execução em outras arquiteturas e sistemas operacionais (já que o *Exchange 2000 Server* [EXC 2001] é fortemente integrado, e dependente, do sistema operacional *Windows 2000 Server* [MIC 2001]). Com a atual ascensão do software livre, pode-se também construir ambientes como WOSDIE, a partir da utilização de ferramentas / aplicativos como componentes, que são construídos com conceitos de software livre e ou código aberto. A integração destes componentes seria feita com PHP, que é um mecanismo de construção de páginas *web* baseado na execução de *scripts* (semelhante ao ASP) e que pode ser executado na maioria dos navegadores existentes, independente de arquitetura e sistema operacional.

- **A utilização de Documentos XML**: utilizar XML para o armazenamento das informações e dados referentes ao WOSDIE é uma excelente maneira de compartilhar os dados com outros ambientes e ferramentas e possibilitar a integração de dados. Várias ferramentas e ambientes atuais são capazes de lidar com documentos XML como meio de disponibilização, armazenamento e compartilhamento de informações. Além disso, XML é uma linguagem "portável", no sentido que permite que aplicativos nativos de diferentes arquiteturas e sistemas operacionais possam trabalhar com informações disponibilizadas neste tipo de documento.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT atividade (content-class, idProjeto,descProjeto, idAtividade, descAtividade, percentual?,
responsavel, dataInicio, duracao, template?, ferramenta?, dataModificacao, observacoes, artefato*,
status?, resultado, hora, duracaoEfetiva, erevisao, guideline, papel)>

<!ELEMENT content-class (#PCDATA)>
<!ELEMENT idProjeto (#PCDATA)>
<!ELEMENT descProjeto (#PCDATA)>
<!ELEMENT idAtividade (#PCDATA)>
<!ELEMENT descAtividade (#PCDATA)>
<!ELEMENT percentual (#PCDATA)>
<!ELEMENT responsavel (#PCDATA)>
<!ELEMENT dataInicio (#PCDATA)>
<!ELEMENT duracao (#PCDATA)>
<!ELEMENT template (#PCDATA)>
<!ELEMENT ferramenta (#PCDATA)>
<!ELEMENT dataModificacao (#PCDATA)>
<!ELEMENT observacoes (#PCDATA)>
<!ELEMENT artefato (#PCDATA)>
<!ELEMENT status (#PCDATA)>
<!ELEMENT resultado (#PCDATA)>
<!ELEMENT hora (#PCDATA)>
<!ELEMENT duracaoEfetiva (#PCDATA)>
<!ELEMENT erevisao (#PCDATA)>
<!ELEMENT guideline (#PCDATA)>
<!ELEMENT papel (#PCDATA)>
```

FIGURA 6.31 - Documento DTD para um item de Atividade

A utilização de documentos XML pelo WOSDIE necessita que documentos DTD (*Document Type Definition - Definição de Tipo de Documento*) sejam definidos para o armazenamento de dados referentes a itens utilizados pelo WOSDIE como atividades, usuários, papéis, ferramentas, etc. (todos os itens citados na seção 6.2). Na Figura 6.31 é mostrado um exemplo de DTD de um item de atividade e na Figura 6.32 um exemplo de documento XML construído com base nesta DTD. Documentos XML e documentos DTD poderiam ser utilizados no WOSDIE, substituindo os itens do *Exchange*, que, por sua vez, são extremamente dependentes de aplicativo e sistema operacional que os acessa (*Exchange 2000 Server* e *Windows 2000 Server*).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE atividade SYSTEM "atividade.dtd">

<atividade>
  <content-class>gpmgt:content-classes:atividade</content-class>
  <idProjeto>{20B4EE57-2E44-4C65-82E1-4DC903D1DA19}.eml</idProjeto>
  <descProjeto>Contas</descProjeto>
  <idAtividade>EliSolInt</idAtividade>
  <descAtividade>Elicitar Solicitação dos Interessados</descAtividade>
  <percentual>0</percentual>
  <responsavel>Jose Silveira</responsavel>
  <dataInicio>20/08/2002</dataInicio>
  <duracao>5</duracao>
  <template>http://Amadeus/RUP/Templates/rup_stkreq.dot</template>
  <ferramenta>Word</ferramenta>
  <dataModificacao>22/08/2002</dataModificacao>
  <observacoes>Sistema será integrado com legado</observacoes>
  <artefato>Elicitacao_Contas.doc</artefato>
  <status></status>
  <resultado></resultado>
  <hora></hora>
  <duracaoEfetiva></duracaoEfetiva>
  <erevisao>Falso</erevisao>
  <guideline>ac_elstk.htm</guideline>
  <papel>Analista de Sistemas</papel>
</atividade>
```

FIGURA 6.32 - Exemplo de Documento XML referente a atividade "Elicitar Solicitação dos Interessados"

6.11 Resumo do Capítulo

Neste capítulo foi realizada a descrição do WOSDIE, que foi o protótipo de experimentação construído durante este trabalho.

O WOSDIE foi implementado utilizando os seguintes componentes:

- O *Exchange 2000 Server* (E2K) como o gerente dos processos do ambiente;
- O *Workflow Designer* como o dispositivo de modelagem para os processos;
- Documentos ASP para a geração de páginas dinâmicas para o acesso dos usuários e integração dos produtos COTS utilizados;
- O *Web Storage System* foi utilizado como um banco de dados que mantém as informações referentes aos processos e documentos associados ao ambiente e aos projetos do ambiente (projetos em execução no WOSDIE);

- Itens do E2K foram utilizados como dispositos de armazenamento das informações do WOSDIE e dos processos;
- Componentes *ActiveX* (automação OLE) foram utilizados para permitir a ativação de ferramentas externas (*LaunchinIE*) e o "*upload*" de arquivos dos computadores clientes para o servidor do WOSDIE (*Upload Dundas*);

Uma avaliação do WOSDIE foi realizada, tomando por base uma grade de avaliação proposta por Ambriola et. al. [AMV 97] (Seção 6.9), e as possíveis extensões para o WOSDIE foram descritas na seção 6.10.

7 Conclusões

Ambientes de Desenvolvimento de Software (ADS) têm sido muito pesquisados e utilizados nestas últimas décadas, vários artigos foram publicados relatando pesquisas sobre arquitetura, implementação, avaliação e "paradigmas" de ADS. Alguns ambientes comerciais foram criados e estão em utilização em diversas organizações de desenvolvimento de software. O desenvolvimento de ambientes deste tipo é uma área interessante de pesquisa, dada a carência ainda existente de ambientes de fácil implantação e baixo custo de treinamento, e que forneça meios de automatização do processo de software de maneira fácil e ao mesmo tempo explícita. A construção de ADS de maneira mais rápida e barata é um importante objetivo para organizações de desenvolvimento de software.

A utilização de produtos comerciais (COTS) na construção de ADS e outros tipos de ambientes é uma alternativa para o problema de construção rápida e barata. No capítulo 5 esta alternativa foi abordada, mostrando as vantagens e desvantagens de utilização de produtos COTS e o que acarreta a utilização destes produtos em termos de arquitetura e implementação.

A utilização de Sistemas de Gerência de Workflow (SGW) como componente na construção de um ambiente de desenvolvimento de software foi apresentada, juntamente com uma proposta de arquitetura que unifica, além de um SGW, um banco de dados, um servidor *Web* e computadores clientes distribuídos através da *internet*. Os SGW permitem a modelagem explícita de processos de negócio, e, como tal, processos de software. A utilização de um paradigma (*Workflow*) de propósito geral forneceu uma maior flexibilidade em relação ao modelo de processo do ambiente construído. A integração de componentes comerciais (COTS) possibilitou que diferentes sistemas (e ferramentas) fossem integrados e/ou ativados. Os SGW permitem a execução distribuída de atividades, podendo inclusive ser executado via *Web*, e permitem também o gerenciamento de recursos humanos dentro de projetos de software (ou mesmo outros domínios), através do assinalamento automático de atividades aos participantes de um projeto de software.

WOSDIE foi implementado sobre o *Exchange 2000 Server* (SGW) e utiliza o *Workflow Designer* como mecanismo de modelagem de processos. A utilização de uma ferramenta de gerência de processos pronta facilitou a construção do protótipo, visto que as funcionalidades existentes neste SGW permitiram ao WOSDIE todas as funcionalidades inerentes a uma ferramenta de gerência de processos.

Documentos ASP permitiram o acesso ao WSS para a recuperação, gravação e atualização de informações do ambiente e também possibilitaram fornecer ao WOSDIE uma “aparência” de página *Web*. Esta característica parece ser uma tendência na atualidade, já que diversos tipos de aplicativos estão sendo, ou já foram, implementados de modo que possam ser executados via internet, permitindo que o acesso a estes aplicativos seja feito da mesma forma que o acesso a uma página *web*. O WOSDIE utiliza itens do *Exchange* (seção 0) para gerar seus dados. Estes itens são criados a partir da biblioteca CDO (*Collaboration Data Objects*) [MIC 2001a] e são

acessados por *scripts* contidos nos documentos ASP. Estes *scripts* geram consultas SQL que são submetidas ao WSS para a recuperação de informações. Uma vantagem associada ao uso da biblioteca CDO e do WSS é a possibilidade de serem consultados via internet. Os documentos ASP (e algumas funções implementadas em *JavaScript*) fizeram o papel de software de integração (ou o termo em inglês *glueware*) dos componentes utilizados para a construção do WOSDIE.

Foi extraído do processo unificado *Rational* (RUP) um exemplo de processo de software que abrange todas as etapas de um ciclo de vida de software, isto é, que abrange todo um processo de construção de um produto de software. Este processo foi adaptado e modelado (conforme a seção 6.3.1) no *Workflow Designer* e foi utilizado como processo padrão na execução de projetos no ambiente WOSDIE. Também foram utilizados documentos e gabaritos definidos pelo processo RUP [RAT 2001]. O RUP é um processo que, segundo seus autores [RAT 2001] [RAT 98] [KRU 2000] [JAC 99], integra as melhores práticas para desenvolvimento de software e vem sendo muito utilizado, tanto a nível acadêmico quanto industrial.

A seção 6.3.1 mostrou como realizar a modelagem de um processo de software na ferramenta *Workflow Designer*, atividade que utilizou muito tempo de implementação e necessita ser realizada seguindo certos passos bem definidos para que possa ser bem executada.

A seção 6.4 mostrou algumas possibilidades para a ativação de ferramentas externas a partir de um navegador *Web*, ação difícil de ser implementada devido as restrições de segurança existentes no ambiente internet e inerente aos navegadores *Web*.

A construção de um ADS baseada na arquitetura definida no Capítulo 5 mostrou que é possível se gerir as atividades associadas ao desenvolvimento de software, através da modelagem e execução de processos de software em um SGW, permitindo que os profissionais envolvidos em um projeto de software conheçam quais tarefas devem executar (pois o próprio ambiente indica quais são estas tarefas, através da criação de listas de trabalho associadas aos usuários), que ferramentas utilizar para a execução das suas tarefas (visto que, associadas as tarefas dos participantes de um projeto, também estão associadas as relativas ferramentas), como executar estas tarefas (o ambiente se encarrega de fornecer informações (extraídas do processo RUP) detalhadas a respeito das tarefas), e que também conheçam como acessar artefatos e informações associados a um projeto de software (o ambiente mantém os artefatos e informações - como documentos de arquitetura de software, casos de uso, resultados de revisões, resultados de testes, etc. - associadas aos projetos e permite o acesso aos mesmos).

Duas avaliações foram realizadas neste trabalho:

1. Uma avaliação da arquitetura proposta na seção 5.4 - que foi baseada nos requisitos necessários para um Ambiente de Desenvolvimento de Software (ADS) - definidos na seção 2.4.
2. Uma avaliação do protótipo construído (WOSDIE) na seção 6.9, que foi baseado na grade de avaliação de ADS (mais especificamente para ADS-CP – Ambientes de Desenvolvimento de Software Centrados em Processo) proposta por Ambriola et. al. [AMV 97] (a grade também é descrita na seção 6.9).

O emprego de produtos comerciais (COTS) como componentes de construção permitiu a construção rápida do protótipo WOSDIE; e mais funcionalidades podem ser integradas ao WOSDIE com pouca, ou nenhuma, necessidade de modificação do software de integração (documentos ASP) que agrupou os componentes em um ambiente único (WOSDIE). A possibilidade de se integrar produtos já existentes nas organizações é uma vantagem clara na construção de ADS, principalmente em pequenas organizações, visto que os custos de desenvolvimento de ambientes como ADS são considerados altos.

A utilização de SGW como mecanismo de gerência de processos no WOSDIE proporcionou que o processo de software extraído do RUP (descrito na seção 6.3) fosse modelado e executado. A modelagem de processos em um SGW mostrou ser muito flexível e generalizada, pois os dispositivos de modelagem dos SGW permitem que um modelo de processo seja modelado na granularidade necessária, isto é, possibilita que as atividades sejam descritas tanto em um nível detalhado (por exemplo: construir o modelo de casos de uso) quanto em um nível mais abstrato (por exemplo: realizar a análise dos requisitos de software). A capacidade de poderem ser utilizados em ambientes baseados na internet é outro ponto a favor da utilização de SGW, já que aplicações Web são cada vez mais requisitadas por engenheiros e desenvolvedores de software e pelas pessoas em geral, dentro dos mais variados domínios de aplicação.

O processo RUP se mostrou adaptável e também aplicável em um nível simples de desenvolvimento de software, isto é, poderia ser aplicado por pequenas organizações de desenvolvimento de software, bastanto se adaptar o processo. Foi possível extrair do RUP o modelo de processo descrito na seção 6.3 e utilizar os artefatos e manuais de orientação (guidelines) propostos neste processo. Pode-se afirmar que RUP - apesar de ser um processo complexo, completo, abrangente e que apresenta elementos que nem mesmo organizações de desenvolvimento de grande porte utilizam - pode ser utilizado por organizações de qualquer tamanho, variando desde pequenas “software houses” até grandes organizações de desenvolvimento de software que possuem filiais em vários países.

A construção de ADS é um problema sério dentro da área de Engenharia de Software. Construir um ambiente que irá auxiliar grupos de pessoas no desenvolvimento de produtos de software é uma tarefa complexa. Novos processos e metodologias de desenvolvimento de software são propostos periodicamente, e os já existentes são atualizados em períodos de tempo cada vez mais curtos. Um ADS deve ser extensível de maneira que:

- Possibilite que um modelo de processo de software possa ser modelado e executado através de algum mecanismo de *motor de processo* (Sistema de Gerência de Workflow – SGW) e que este modelo de processo possa ser modificado de acordo com a experiência obtida sobre aquele processo ao longo de sua utilização e aplicação;
- Novas ferramentas possam ser integradas ao ambiente, para que o ambiente possa ser atualizado em relação a novas ferramentas e metodologias (disponibilizadas através das ferramentas), e que ferramentas que já existem nas organizações possam ser reaproveitadas pela sua integração ao ADS;
- O ambiente possa ser configurado em relação a sua base de informações sobre as atividades realizadas no processo de desenvolvimento de software,

os gabaritos disponíveis, informações sobre os modelos de processo utilizados, papéis desempenhados, ferramentas integradas, etc., isto é, deve possibilitar uma Gerência de Configuração do Ambiente.

Os tópicos acima motivam ainda mais a utilização de abordagens de construção baseadas na integração de componentes comerciais de prateleira (COTS). Esta abordagem de construção possibilita que o ADS resultante possa ser mais facilmente estendido e melhorado. A extensão e melhoramento de ADS têm associado, possivelmente, o custo de modificação do software *glueware* que integra o ADS.

TABELA 7.1 – Quadro Comparativo entre as Abordagens de Construção utilizando Produtos COTS e Construção a partir do Zero

Característica Analisada	1. Abordagem de Integração de Produtos COTS	2. Abordagem de Construção a partir do zero
Qualidade e Performance	**Vantagens**: o ADS provê ferramentas avançadas de desenvolvimento de software, são produtos normalmente já bem utilizados, portanto, mais confiáveis. **Desvantagens**: necessita de *glueware* para a integração dos componentes, isto pode afetar a qualidade e performance, devido à existência de produtos mal documentados e com interfaces proprietárias.	**Vantagens**: o ADS, por ser construído por completo, provavelmente não apresentará problemas de incompatibilidade entre seus módulos e terão uma melhor performance e qualidade. A qualidade do ADS vai depender do processo de desenvolvimento. **Desvantagens**: o processo de desenvolvimento do ADS deve ser bem definido, para que se obtenha um ADS que contemple os seus requisitos de qualidade e performance.
Tempo de Construção e Cronograma	**Vantagens**: tempo de construção reduzido, necessitando, no entanto, de construção de software de integração (*glueware*). **Desvantagens**: dependendo da seleção dos produtos COTS utilizados, o tempo gasto para integrá-los com *glueware* pode ser grande.	**Vantagens**: o produto é construído baseado em requisitos específicos. **Desvantagens**: maior tempo de desenvolvimento, todas as funcionalidades do ADS devem ser implementadas.
Manutenção	**Vantagens**: componentes podem ser modificados e/ou atualizados com certa facilidade. Eliminação de erros só é realizada em cima de software de integração. **Desvantagens**: a atualização dos produtos COTS acarreta custos de atualização de versões.	**Vantagens**: - **Desvantagens**: o código fonte deve ser modificado, ocasionando alocação de recursos para a tarefa.
Custo de Construção	**Vantagens**: custo reduzido em comparação à outra abordagem e reuso de ferramentas já existentes. **Desvantagens**: custos associados à seleção, integração e licenciamento de produtos COTS.	**Vantagens**: ADS construído segundo requisitos específicos. **Desvantagens**: custo de construção elevado devido à complexidade, tamanho e funcionalidade de um ADS.
Extensibilidade	**Vantagens**: fácil integração de novas ferramentas e/ou recursos. **Desvantagens**: possível necessidade de adaptação do ADS (*glueware*).	**Vantagens**: - **Desvantagens**: não é estendível, é necessário que uma nova versão do ADS seja liberada.
Tamanho e Complexidade	**Vantagens**: ADS mais avançados podem ser construídos com mais facilidade. **Desvantagens**: perda de controle sobre os componentes (para componentes mal documentados e com interfaces fora dos padrões) e adaptação do software de integração.	**Vantagens**: -. **Desvantagens**: quanto maior a complexidade e funcionalidade de um ADS, mais demorado, caro e arriscado é o esforço de desenvolvimento do mesmo.
Estado da Arte	**Vantagens**: produtos COTS são tecnologia de ponta, pois, normalmente, seus fornecedores são especialistas na construção daquele produto. **Desvantagens**: -.	**Vantagens**: - **Desvantagens**: o tempo de construção pode tornar o ADS obsoleto.

A Tabela 7.1 mostra um quadro comparativo, com base em algumas características consideradas importantes, entre duas abordagens de construção de ADS:

(1) Utilizando produtos COTS como componentes de construção - incluindo entre estes componentes: Sistemas de Gerência de Workflow (motor de

processo), Sistemas de Banco de Dados (repositório), e outros – e software de integração para agrupar os componentes em um ADS;

(2) Construindo o ADS a partir do zero, implementando todas as funcionalidades do ambiente.

O *Microsoft Exchange 2000 Server* (E2K) possui características interessantes, que são:

- A possibilidade de utilização dos itens do *Exchange* (através da biblioteca CDO), que permitem criação e edição de objetos com propriedades definidas pelo programador do *workflow* (podem ser criadas propriedades específicas para cada tipo de aplicação) e que podem ser definidas de acordo com o necessário, tornando a tarefa de modelagem dos processos de *workflow* mais adaptável às aplicações e aos conceitos associados;
- A forte integração com o sistema operacional *Windows 2000 Server* e com o *Web Storage System* (WSS), que permitem fácil acesso e compartilhamento de dados (heterogêneos) relativos às aplicações do E2K.
- A possibilidade de se fazer consultas aos dados das aplicações do E2K através da linguagem de consulta SQL embutidas em código ASP e utilizando as bibliotecas ADO e WebDAV;

O E2K demonstrou ser um SGW de fraca performance, seu tempo de resposta para requisições é lento, mesmo para acesso em rede local, principalmente em situações de transição de estados no modelo de processo. Outros pontos fracos do E2K são:

- A necessidade de programação de serviços básicos que deveriam ser fornecidos por um SGW ideal, são estes: ativação de ferramentas externas de acordo com a atividade a ser executada, envio de mensagens eletrônicas para os participantes do ambiente, geração de itens de atividades e outros, geração de páginas/formulários HTML (através de documentos ASP) para a edição de atividades, configuração de papéis, ferramentas e atividades, visualização de listas de trabalhos (*Work Lists - to do Lists*), de membros da equipe e situação dos projetos, definição dos responsáveis pelas atividades, e outros;
- O excessivo trabalho de configuração e criação de pastas para as aplicações do E2K, definição de propriedades e direitos de acesso sobre o *workflow*, falta de dispositivos de depuração do *Workflow Designer* para permiter melhor modelagem dos processos, etc.

7.1 Trabalhos Futuros

Alguns trabalhos futuros podem ser propostos a partir deste trabalho:

- Disponibilizar algum mecanismo de interação entre os usuários do protótipo, como um sistema de *Chat*, para viabilizar a realização de reuniões, revisões, etc. sem a necessidade dos desenvolvedores se encontrarem em um local específico,

possibilitando que este tipo de reunião possa ser realizado via *Internet*, através do próprio ambiente.

- Integrar ao WOSDIE ferramentas colaborativas para edição de artefatos de software, como a proposta por Cayres [CAY 99] e outros. Este tipo de ferramenta permite que vários desenvolvedores editem um artefato de maneira colaborativa, cada um trabalhando sobre o artefato e propondo modificações. O responsável pelo artefato, ao final da edição do artefato, aceita ou não as alterações e propostas dos demais construtores do artefato.
- Adicionar ao WOSDIE um dispositivo de coleta de métricas. Os *scripts* implementados nos documentos ASP que formam a interface do WOSDIE podem ser modificados para, coletarem informações relevantes sobre o ambiente, projetos e usuários, de modo que possam ser utilizadas, após algum processamento estatístico, como dados para a realização de medidas em relação a utilização do WOSDIE.
- Modificar o WOSDIE de modo que sejam utilizados documentos XML, ao invés de itens do Exchange (itens criados a partir da biblioteca CDO [MSD01]), para o armazenamento das informações do ambiente. Um exemplo de *script* (em *VBScript* - ASP) para a criação de um documento XML é mostrado abaixo (Figura 7.1).

```
Function saveToXml(strXMLFilePath, strFileName)
'Declaracao de variaveis locais.
Dim objDom, objRoot, objRecord, objPI, blnFileExists
        ' Instanciação do Microsoft XMLDOM.
Set objDom = server.CreateObject("Microsoft.XMLDOM")
objDom.preserveWhiteSpace = True
  blnFileExists = objDom.Load(strXMLFilePath & "\" & strFileName)
If blnFileExists = True Then
        ' Se o arquivo foi carregado com sucesso, o objeto raiz (root) do doc. XML é setado
 Set objRoot = objDom.documentElement
Else
        ' Criação do objeto raiz e adição do mesmo para o documento XML.
 Set objRoot = objDom.createElement("Atividade")
 objDom.appendChild objRoot
End If
        ' Criação de um novo elemento, neste caso o elemento identificador de um projeto.
Set objRecord = objDom.createElement("idProjeto")
objRecord.Text = workflowSession.Fields("DAV:displayname").Value
objRoot.appendChild objRecord
        ' Criação de um novo elemento, neste caso o elemento descritor de um projeto.
objRecord = objDom.createElement("descProjeto")
objRecord.Text = workflowSession.Fields("gpmgt:DescProjeto").Value
objRoot.appendChild objRecord
  '... O restante dos elementos são criados aqui.
If blnFileExists = False then
 'Se é um arquivo novo, é necessário inserir as instruções de processamento XML.
 Set objPI = objDom.createProcessingInstruction("xml", "version='1.0' encoding='ISO-8859-1'")
 objDom.insertBefore objPI, objDom.childNodes(0)
End If
'Salvando o documento XML.
objDom.save strXMLFilePath & "\" & strFileName

End Function
```

FIGURA 7.1 - Exemplo de *Script* para criação de documento XML

- Integrar ao WOSDIE uma ferramenta de Gerência de Projetos, com o intuito de possibilitar uma melhor definição e edição de cronogramas de projeto, e também para a atualização de cronogramas de acordo com a execução do processo e possíveis atrasos.

7.2 Considerações Finais

Uma experimentação no nível de construção de ADS foi realizada neste trabalho, e toda a parte conceitual e prática necessária para esta construção foi abordada. Foram utilizados vários componentes e ferramentas para a geração do WOSDIE, alguns disponibilizados comercialmente (como *E2K*, *RUP*, *Microsoft Word*) e outros disponibilizados gratuitamente via internet (como os componentes *Active X LaunchinIE* e *Upload Dundas*), demonstrando a possibilidade de integração de qualquer tipo de componente para a construção dos mais variados ambientes aplicativos.

Foram mostrados pontos de extensões e propostos trabalhos futuros a serem realizados sobre o mesmo tema abordado neste trabalho e temas correlatos.

Os dispositivos e métodos de implementação necessários para a implementação do protótipo deste trabalho foram descritos, definindo-se o que foi necessário para a implementação e como esta implementação foi realizada.

Referências

[AMA 2002] PROJETO AMADEUS. **Ambientes e Metodologias Adaptáveis de DEsenvolvimento Unificado de Software**. Disponível em: <http://www.inf.ufrgs.br/amadeus/>. Acesso em: 20 fev. 2002.

[AMB 2001] AMBLER, Scott W. **Enterprise Unified Process:** Enhancing the Unified Process to Meet the Real-World Needs of Your Organization. Ronin International White Paper. Nov. 2001. Disponível em: <http://www.ronin-intl.com/publications/unifiedProcess.PDF>. Acesso em: 06 fev. 2002.

[AMB 98] AMBLER, Scott W. **An Introduction to Process Patterns**. Ambysoft Inc. Whitepaper. June 2001. Disponível em: <http://www.ambysoft.com/processPatternsPaper.html>. Acesso em: 05 fev. 2002.

[AMV 97] AMBRIOLA, V.; CONRADI, R.; FUGGETA, A. Assessing Process-Centered Software Engineering Environments. **ACM Transactions on Software Engineering and Methodology**, New York, v. 6, n. 3, p. 283-328, July 1997.

[ARA 99] ARAÚJO, Renata M.; BORGES, Marcos R. S. Sobre a Aplicabilidade de Sistemas de Workflow no Suporte a Processos de Software. In: JORNADAS IBEROAMERICANAS DE INGENIERIA DE REQUISITOS Y AMBIENTES DE SOFTWARE, IDEAS, 2., 1999, Alajuela, Costa Rica. **Memorias**. Cartago: CIT, 1999.

[ARC 2001] PROJETO Arcadia. Irvine: Department of Information and Computer Science, University of California. Disponível em: <http://www.ics.uci.edu/~arcadia/>. Acesso em : 15 maio 2001.

[BAN 96] BANDINELLI, S.; DI NITTO, E.; FUGGETTA, A. Supporting Cooperation in the SPADE-1 Environment. **ACM Transactions on Software Engineering**, New York, v. 22, n. 12, Dec. 1996.

[BAR 2000] BARNES, A.; GRAY, J. COTS, Workflow, and Software Process Management: An Exploration of Software Engineering Tool Development. In: AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE, 2000, Australia. **Proceedings…** Los Alamitos: IEEE, 2000.

[BAR 2000a] BARNES, A.; GRAY, J. Workflow Products as a Tool Construction Technology for Process-centered SEE. In: INTERNATIONAL CONFERENCE ON SOFTWARE METHODS AND TOOLS, SMT, 2000, Australia. **Proceedings…** Los Alamitos: IEEE, 2000.

[BEN 94] BEN-SHAUL, I.; KAISER, G. A Paradigm for Decentralized Process Modeling and its Realization in the Oz Environment. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING,

ICSE, 16., 1994, Sorrento, Italy. **Proceedings…** Los Alamitos: IEEE, 1994.

[BIZ 2000] HANDYSOFT CORPORATION. **Welcome to BizFlow - 2000**. Página da HandySoft - Produtos/BizFlow. Disponível em: <http://www.handysoft.com/products/bizflow/index.html>. Acesso em: 19 dez. 2000.

[BOO 2000] BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML:** guia do usuário. Tradução de Fábio Freitas da Silva. Rio de Janeiro: Campus, 2000.

[BOO 99] BOOCH, G. et al. **UML for XML Schema Mapping Specification**. Rational Website, December 1999. Disponível em: <http://www.rational.com/media/uml/resources/media/uml_xmlschema33.pdf>. Acesso em: 20 jan. 2003. (Technical report).

[BOR 2002] BORLAND SOFTWARE CORPORATION. **Borland Home Page**. Disponível em: <http://www.borland.com>. Acesso em: 28 fev. 2002.

[BRO 93] BROWN, Alan W. An Examination of the Current State of IPSE Technology. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 15., 1993, Baltimore, MA, USA. **Proceedings…** Los Alamitos: IEEE, 1993.

[BUS 98] BUSSLER, Christoph. Workflow Instance Scheduling with Project Management Tools. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, DEXA, 9., 1998, Viena. **Database and Expert Systems Applications**: proceedings. Berlin: Springer-Verlag, 1998.

[CAL 96] CALLAHAN, J.; RAMAKRISHNAM, S. Software Project Management and Measurement on the World-Wide-Web (WWW). In: WORKSHOP ON ENABLING TECHNOLOGIES: INFRASTRUCTURE FOR COLLABORATIVE ENTERPRISES, WETICE, 5., 1996, Stanford, CA, USA. **Proceedings…** Los Alamitos: IEEE, 1996.

[CAR 97] CARNEY, David. **Assembling Large Systems from COTS Components:** Opportunities, Cautions, and Complexities. SEI – Software Engineering Institute, June 1997. Disponível em: <http://www.sei.cmu.edu/cbs/papers/monographs/assembling-systems/assembling.systems.htm>. Acesso em: 14 dez. 2001.

[CAY 99] CAYRES, Paulo Henrique. Um Editor Diagramático Distribuído para Modelagem de Sistemas. In: SEMANA ACADÊMICA DO PPGC, 4., 1999, Porto Alegre. **Anais ...** Porto Alegre: PPGC da UFRGS, 1999. p. 207-210.

[CHA 97] CHAN, Daniel K. C.; LEUNG, Karl R.P.H. A Workflow Vista of the Software Process. In: INTERNATIONAL WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATIONS, DEXA, 8., 1997, Toulouse, France. **Database and Expert Systems Applications**: proceedings. Berlin: Springer-Verlag, 1997.

[CHA 97a] CHAN, Daniel K. C.; LEUNG, Karl R.P.H. Software Development as a Workflow Process. In: ASIA PACIFIC SOFTWARE ENGINEERING

CONFERENCE AND INTERNATIONAL COMPUTER SCIENCE CONFERENCE, APSEC / ICSC, 1997, Hong Kong, SAR, China. **Proceedings…** Los Alamitos: IEEE, 1997.

[CHA 99] CHAN, Daniel K. C. Form Management in the Vicomte Workflow System. In: INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, HICSS, 32., 1999, Maui, Hawaii. **Proceedings...** [S.l.: s.n.], 1999.

[CHR 99] CHRISTIE, Alan M. Simulation in Support of CMM-based Process Improvement. **The Journal of Systems and Software**, [S.l.], v. 46, n. 2/3, p. 107-112, 1999.

[CHU 2002] CHUNG, Lawrence; COOPER, Kendra. A Knowledge-based COTS-Aware Requirements Engineering Approach. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, SEKE, 14., 2002, Ischia, Italy. **Proceedings...** New York: ACM, 2002.

[COJ 99] CONALLEN, Jim. **Modeling Web Application Architectures with UML.** Rational Software White Paper. June 1999. Disponível em: <http://www.rational.com/products/whitepapers/100462.jsp>. Acesso em: 08 dez. 2001.

[CON 2000] CONRAD, R.; SCHEFFNER, D. **XML Conceptual Modeling using UML**. Berlin: Institute of computer science, 2000. Extended version. (Technical report).

[COR 93] CONRADI, R.; FERNSTROM, C.; FUGGETTA, A. A Conceptual Framework for Evolving Software Processes. **ACM SIGSOFT Software Engineering Notes**, [S.l.], v. 18, n. 4, p. 26-34, Oct. 1993. Disponível em: <http://citeseer.nj.nec.com/conradi93conceptual.html>. Acesso em: 12 dez. 2001.

[DAV 2000] DAVIS, Jonh S. **IBM MQSeries Workflow:** Staff Assignment Techniques. IBM Corp., July 2000. Disponível em: <http://www-4.ibm.com/software/ts/mqseries/Workflow/supphumn.pdf>. Acesso em: 22 nov. 2000.

[DIC 97] DICATERINO, A. et al. **An Introduction to Workflow Management Systems.** Albany, USA: University of Albany, Center for Technology in Government, Nov. 1997. Disponível em: <http://www.ctg.albany.edu/resources/pdfrpwp/mfa-2.pdf>. Acesso em: 20 fev. 2002.

[DUN 2002] DUNDAS SOFTWARE LTD. **Dundas Upload**. Disponível em: <http://www.dundas.com/>. Acesso em: 11 mar. 2002.

[EPO 2001] EPOS Group Home Page. Disponível em: <http://www.idi.ntnu.no/~epos/>. Acesso em: 16 maio 2001.

[ERI 2000] ERICSSON, Maria. **Developing Large-scale Systems with the Rational Unified Process.** Rational Software Whitepaper. 2000. Disponível em: <http://www.rational.com/products/whitepapers/sis.jsp>. Acesso em: 04 dez. 2001.

[EUP 2003] ENTERPRISE UNIFIED PROCESS (EUP). **EUP Home Page**. Disponível em: <http://www.enterpriseunifiedprocess.info/>. Acesso em: 08 jan. 2003.

[EXC 2001] MICROSOFT CORPORATION. **Exchange Server Home**. Disponível em: <http://www.microsoft.com/exchange/default.htm>. Acesso em: 11 maio 2001.

[FIN 94] FINKELSTEIN, A.; KRAMER, J.; NUSEIBECH, B.(Ed.). **Software Process Modelling and Technology**. Chichester, UK: John Wiley & Sons: Research Studies Press, 1994. 362 p.

[FOW 2000] FOWLER, Martin. **UML Essencial :** um breve guia para a linguagem-padrão de modelagem de objetos. Porto Alegre: Bookman, 2000. 169 p.

[FUG 2000] FUGGETTA, Alfonso. Software Process: a roadmap. In: FINKELSTEIN, A. (Ed.). **The Future of Software Engineering**. New York, USA: ACM Press, 2000. Special Volume published in conjuction with ICSE 2000, Limerick, Ireland.

[FUG 96] FUGGETTA, A. Functionality and Architecture of PSEE. **Information and Software Technology**, [S.l.], v. 38, p. 289-293, 1996.

[GAL 2000] GALEA, Nick. **Workflow technology - an introduction.** GFI Communications Ltd. Disponível em: <http://www.workflowsoftware.com/workflowwp.pdf>. Acesso em: 21 ago. 2000.

[GAM 2000] GAMMA, Erich et al. **Padrões de Projeto:** soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2000. 364 p.

[GAR 94] GARG, P.K. et al. The SMART Approach for Software Process Engineering. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 16., 1994, Sorrento, Italy. **Proceedings…** Los Alamitos: IEEE, 1994.

[GEO 95] GEORGAKOPOULOS, Dimitrios; HORNICK, M. **An Overview of Workflow Management:** From Process Modeling to Workflow Automation Infrastructure. GTE Laboratories, Distributed and Parallel Databases, Apr. 1995. Disponível em: <ftp://ftp.gte.com/pub/dom/reports/GEOR95a.ps>. Acesso em: 23 dez. 2000.

[GIM 94] GIMENES, Itana M. S. **O Processo de Engenharia de Software:** ambientes e formalismos. Caxambú, MG, Brasil: Sociedade Brasileira de Computação, 1994.

[GIM 99] GIMENES, Itana M. S.; WEISS, Gerson M.; HUZITA, Elisa H. M. Um Padrão para Definição de um Gerenciador de Processos de Software. In: JORNADAS IBEROAMERICANAS DE INGENIERIA DE REQUISITOS Y AMBIENTES DE SOFTWARE, IDEAS, 2., 1999, Alajuela, Costa Rica. **Memórias**. Cartago: CIT, 1999.

[GRU 2000] GRUNDY, J. et al. Constructing Component-based Software Engineering Environments: issues and experiences. **Information and Software Technology**, [S.l.], v. 42, p. 103-114, 2000.

[HIT 98] HITOMI, Arthur S.; LE, Dong. Endeavors and Component Reuse in Web-Driven Process Workflow. In: CALIFORNIA SOFTWARE SYMPOSIUM, October 1998, Irvine, CA, USA. **Proceedings…** Disponível em: <http://www.ics.uci.edu/pub/endeavors/docs/papers/>. Acesso em: 16 ago. 2001.

[HOL 95] HOLLINGSWORTH, David. **The Workflow Reference Model.** The Workflow Managment Coalition Specification. Jan. 1995. Disponível em: <http://www.wfmc.org/standards/docs/tc003v11.pdf>. Acesso em: 31 ago. 2000.

[JAC 99] JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **The Unified Software Development Process**. Reading, MA, USA: Addison-Wesley, 1999.

[JAR 94] JARKE, Matthias; NISSEN, Hans; POHL, Klaus. Tool Integration in Evolving Information Systems Environment. In: GI WORKSHOP INFORMATION SYSTEMS AND ARTIFICIAL INTELLIGENCE: ADMINISTRATION AND PROCESSING OF COMPLEX STRUCTURES, 3., 1994, Hamburg, Germany. **Proceedings…** [S.l.:s.n.], 1994.

[KRU 2000] KRUCHTEN, P. **The Rational Unified Process – An Introduction**. 2nd ed. Reading, MA, USA: Addison-Wesley, 2000.

[LAA 2001] LAAHS, Kevin. **Exchange 2000 Workflow Applications**. Windows 2000 Magazine, Applied Microsoft Technologies Group at Compaq (Focus). Feb. 2001. Disponível em: <http://www.win2000mag.com/Articles/Print.cfm?ArticleID=16424>. Acesso em: 03 Out. 2001.

[LAR 2000] LARMAN, Craig. **Utilizando UML e Padrões:** uma introdução à análise e ao projeto orientados a objetos. Porto Alegre: Bookman, 2000. 492 p.

[LEN 2002] LENZERINI, Maurizio. Data integration: a theoretical perspective. In: ACM SIGMOD-SIGACT-SIGART SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS, PODS, 21., 2002, Madison, Wisconsin, USA. **Proceedings...** New York: ACM, 2002.

[LIM 2000] LIMA, Flávio A. **WIDe:** uma Extensão à UML para Modelagem de Sistemas de Informação na Internet Baseados em Documentos. 2000. 148p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[LOO 99] LOONEY, Michael; BRIGGS, Jim. Some experiences and comments on the use of COTS Software in UK Naval Combat Systems. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 21., 1999, Los Angeles, CA, USA. **Proceedings…** Los Alamitos: IEEE, 1999.

[MAD 90] MADHAVJI, N. et al. Prism=Methodology + Process-oriented Environment. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 12., 1990, Nice, France. **Proceedings...** Los Alamitos: IEEE, 1990.

[MAN 2001] MANZONI, Lisandra. **Uso de Sistema de Gerência de Workflow para Apoiar o Desenvolvimento de Software Baseado no Processo Unificado da Rational Estendido para Alcançar Níveis 2 e 3 do Modelo de Maturidade**. 2001. 202p. Dissertação (Mestrado em Informática) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[MAN 2001a] MANZONI, L.; PRICE, R. T. Identifying Extensions Required by RUP (Rational Unified Process) to Comply with CMM (Capability Maturity Model) Levels 2 and 3. In: SIMPOSIO ARGENTINO EM INGENIERÍA DE SOFTWARE, JAIIO, 2001, Buenos Aires, Argentina. **Anales…** Buenos Aires: [s.n.], 2001. v. 2, p.135-144.

[MAR 2000] MARTIN, M. **Programming Collaborative Web Applications With Microsoft Exchange 2000 Server**. Redmond, WA, USA: Microsoft Press, 2000.

[MAU 99] MAURER, F. et al. Software Process Support over the Internet. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 21., 1999, Los Angeles, CA, USA. **Proceedings…** Los Alamitos: IEEE, 1999.

[MEH 2000] MEHTA, Alok; HEINEMAN, George T. A Framework for COTS Integration and Extension. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 22., 2000, Limerick Ireland. **Proceedings…** Los Alamitos: IEEE, 2000.

[MIC 2000] MICROSOFT CORPORATION. **Connecting People with Knowledge**. Product Overview Document, 2000.

[MIC 2000a] MICROSOFT CORPORATION. **The Business Value of Microsoft Exchange 2000 Server**. Product Overview Document, 2000.

[MIC 2001] MICROSOFT CORPORATION. **Microsoft Home Page**. Disponível em: <http://www.microsoft.com/>. Acesso em: 10 set. 2001.

[MIC 2001a] MICROSOFT CORPORATION. **MSDN Library Page**. Disponível em: <http://msdn.microsoft.com/library/default.asp>. Acesso em: 10 set. 2001.

[MOR 2000] MORISIO, M. et al. Investigating and Improving a COTS-based Software Development. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 22., 2000, Limerick Ireland. **Proceedings…** Los Alamitos: IEEE, 2000.

[MQS 2000] IBM CORPORATION. **MQSeries Workflow Product Introduction.** Disponível em: <http://www-4.ibm.com/software/ts/mqseries/library/brochures/mqswf/mqstech3.pdf>. Acesso em: 04 dez. 2000.

[MWY 2000] MARRA, Marci; WHITCOMB, Valerie; YODER, Doug. **Workflow Designer for Exchange:** Automating Workflow on Exchange Folders, an End-to-End Developer Walkthrough. Redmond, WA, USA: Microsoft Corporation, 2000.

[NOT 2000] NOTARI, D. **Uma Enciclopédia de Ambientes Distribuídos de Desenvolvimento de Software**. 2000. 127p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[OBE 98] OBEMDORF, Patricia. **COTS and Open Systems**. SEI – Software Engineering Institute, Feb. 1998. Disponível em: <http://www.sei.cmu.edu/cbs/papers/monographs/cots-open-systems/cots.open.systems.htm>. Acesso em: 16 dez. 2001.

[OCA 98] OCAMPO, C.; BOTELLA, P. **Some Reflections on Applying Workflow Technology to Software Process**. Barcelona: Departament de Llenguatges i Sistemes Informàtics, 1998. (Internal Report LSI-98-5-R). Disponível em: <http://www.lsi.upc.es/~cocampo/publications.html>. Acesso em: 30 jan. 2002.

[OIN 99] OINAS-KUKKONEN, Harri; ROSSI, Gustavo. On Two Approaches to Software Repositories and Hypertext Functionality. **Journal of Digital Information**, [S.l.], v. 1, n. 4, Jan. 1999. Disponível em: <http://jodi.ecs.soton.ac.uk/Articles/v01/i04/Oinas-Kukkonen/>. Acesso em: 23 jan. 2002.

[OLE 2002] THE WEBMASTER'S REFERENCE LIBRARY. **OLE Automation in JavaScript**. Disponível em: <http://www.webreference.com/js/column55/>. Acesso em: 01 mar. 2002.

[OPE 2002] THE OPEN Website. Disponível em: <http://www.open.org.au>. Acesso em: 04 fev. 2002.

[ORA 98] ORACLE CORPORATION. **Oracle Workflow Cartridge.** Redwood, CA, USA, Nov. 1998. Disponível em: <http://www.atloaug.org/presentations/nov98Workflow.pdf>. Acesso em: 23 nov. 2000.

[ORT 95] ORTIGOSA, Alvaro M. **Proposta de um Ambiente Adaptável de Apoio ao Processo de Desenvolvimento de Software**. 1995. 192p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[PAU 93] PAULK, M. et al. **Key Practices of Capability Maturity Model for Software, Version 1.1**. Pittsburgh, PA: Software Engineering Institute (SEI), 1993. (CMU/SEI-93-TR-25).

[PAY 99] PAYTON, J.; KESHAV, R.; GAMBLE, R. System Development Using the Integrating Component Architectures Process. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 21., 1999, Los Angeles, CA, USA. **Proceedings…** Los Alamitos: IEEE, 1999.

[PER 92] PERIN, Marcelo G. **Um Sistema de Gerenciamento de Hiperdocumentos para Ambientes de Desenvolvimento de Software**. 1992. 173p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[PRO 2002] THE PROCESS Patterns Resource Page. Disponível em: <http://www.ambysoft.com/processPatternsPage.html>. Acesso em: 05 fev. 2002.

[RAT 2001] RATIONAL CORPORATION. **Rational Unified Process 2001**. Disponível em: <http://www.rational.com/>. Acesso em: 26 fev. 2002.

[RAT 98] RATIONAL CORPORATION. **Rational Unified Process:** Best Practices for Software Development Teams. Whitepaper, 1998. Disponível em: <http://www.rational.com/products/whitepapers/100420.jsp>. Acesso em: 14 maio 2001.

[RIZ 2000] RIZZO, T. **Programming Microsoft Outlook and Microsoft Exchange 2000 Server**. 2nd ed. Redmond, WA, USA: Microsoft Press, 2000.

[RON 2003] RONIN INTERNATIONAL, INC. **Ronin International Home Page**. Disponível em: <http://www.ronin-intl.com/>. Acesso em: 09 jan. 2003.

[SCH 97] SCHREYJAK, Stefan. Coupling of Workflow and Component-Oriented Systems. In: INTERNATIONAL WORKSHOP ON COMPONENT-ORIENTED PROGRAMMING, WCOP, 2., 1997, Jyväskylä, Finnland. **Proceedings...** Disponível em: <http://www.informatik.uni-stuttgart.de/ipvr/as/publikationen/StsWCOP97.html>. Acesso em: 21 out. 2000.

[SCH 99] SCHMIDT, Marc-Thomas. The Evolution of Workflow Standards. **IEEE Concurrency**, Los Alamitos, CA, v. 7, n. 3, p. 44-52, July-Sept. 1999.

[SEI 2002] SOFTWARE ENGINEERING INSTITUTE (SEI). Disponível em: <http://www.sei.cmu.edu/>. Acesso em: 20 fev. 2002.

[SIL 99] SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 3. ed. São Paulo: Makron Books, 1999. 778 p.

[TIB 2000] TIBCO SOFTWARE INC. **TIB/InConcert.** Disponível em: <http://www.tibco.com/products/in_concert/index.html>. Acesso em: 09 dez. 2000.

[TOG 2001] TOGETHERSOFT. **TogetherSoft Home Page**. Disponível em: <http://www.togethersoft.com/>. Acesso em: 09 maio 2001.

[ULT 2000] ULTIMUS, INC. **Benefits of Workflow**. Disponível em: <http://www.ultimus.com/ultwf/ultwf.htm>. Acesso em: 24 nov. 2000.

[ULT 2000a] ULTIMUS INC. **Ultimus Workflow Suite**. Disponível em: <http://www.meridian-marketing.com/ULTIMUS/1aframe.html>. Acesso em: 23 nov. 2000.

[VBS 2001] MICROSOFT CORPORATION. **Microsoft Scripting Technologies**. Disponível em: <http://msdn.microsoft.com/scripting/>. Acesso em: 10 set. 2001.

[VEC 2000] VECELLIO, Gary; THOMAS, William M. Issues in the Assurance of Component-Based Software. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 22., 2000, Limerick Ireland. **Proceedings…** Los Alamitos: IEEE, 2000.

[VER 97] VERAART, V. A.; WRIGHT, S.L. Dukas: A Software Process Task Management Enviroment. In: AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE, 1997, Australia. **Proceedings…** Los Alamitos: IEEE, 1997.

[VER 99] VERBYLA, J.; WATTERS, C. Cooperative Hypermedia Management Systems. **Journal of Digital Information**, [S.l.], v. 1, n. 4, Jan. 1999. Disponível em: <http://jodi.ecs.soton.ac.uk/Articles/v01/i04/Verbyla/>. Acesso em: 22 jan. 2002.

[WHI 2002] WHIRLY WIRY WEB, GUIDE TO NEAT CODE. **Open Office from a Web Page**. Disponível em: <http://www.whirlywiryweb.com/article.asp?=%2Fofficedocs&xml=0>. Acesso em: 01 mar. 2002.

[WHI 2002a] WHIRLY WIRY WEB, GUIDE TO NEAT CODE. **Launch-in-IE: Web Pages can Start Applications. Securely**. Disponível em: <http://www.whirlywiryweb.com/article.asp?=%2Flaunchinie&xml=0>. Acesso em: 01 mar. 2002.

[WMC 2001] WORKFLOW MANAGEMENT COALITION. **WFMC Home Page**. Disponível em: <http://www.wfmc.org>. Acesso em: 15 nov. 2001.

[WMC 99] WORKFLOW MANAGEMENT COALITION. **Terminology & Glossary.** Brussels, Belgium, 1999. (Document WFMC TC-1011). Disponível em: <http://www.wfmc.org/standards/docs/glossy3.pdf>. Acesso em: 26 fev. 2002.

[WOR 2000] ULTIMUS, INC. **Workflow, Groupware, and the Role of Ultimus.** Disponível em: <http://www.workflowzone.com/ultwhite/white.htm>. Acesso em: 31 ago. 2000.

[XML 2001] W3C. **Extensible Markup Language** (XML). Disponível em: <http://www.w3.org/XML/>. Acesso em: 16 maio 2001.

[YAK 99] YAKIMOVICH, D.; BIEMAN, J.M.; BASILI, V.R. Software Architecture Classification for Estimating the Cost of COTS Integration. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 21., 1999, Los Angeles, CA, USA. **Proceedings…** Los Alamitos: IEEE, 1999.